European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable D2.4

Virtual Laboratory Integration Report

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
Jacobs University Bremen, IUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universitat Politecnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, CETIM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University of Surrey, UniS, UK
University of Pitesti, UniP, Romania

For more information on this document or the EMANICS Project, please contact:

Dr. Olivier Festor
Technopole de Nancy-Brabois - Campus scientifique
615, rue de Jardin Botanique - B.P. 101
F-54600 Villers Les Nancy Cedex
France
Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title:	Virtual Laboratory Integration Report
Type:	Public
Editor(s):	Jürgen Schönwälder, Ha Manh Tran
E-mail:	j.schoenwaelder@jacobs-university.de
Author(s):	WP2 Partners
Doc ID:	D2.4

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
0.1	2007-07-03	H. Tran, J. Schönwälder	Initial version of a LaTeX template
0.2	2007-10-04	D. Hausheer, C. Morariu, T. Bocek	Added EmanicsLab Details
0.3	2008-01-31	G. Schaffrath	Added SIP4EMANICS architecture details
0.4	2008-02-25	F. Eyermann	Added SIP4EMANICS subsection on MetaVoIP
0.5	2008-03-27	J. Schönwälder, H. Tran, I. Tumar	Added more sections, improved the presentation

Contents

1 Executive Summary

This fourth "Virtual Laboratory Integration Report" presents the activities of the virtual laboratory and common test-beds work package (WP2) in the second phase of the EMANICS project. This phase has started in June 2007 with an open call for the first nine-month period starting from June 2007 to March 2008. With lessons learned from the first phase, the criteria of selecting projects rely more on collaboration among partners, infrastructure and resource exploitations, and face-to-face meetings and exchanges, while keeping work focused on well defined subjects. The first call led to a selection of three projects:

1. The EMANICS distributed computing and storage testbed (EMANICSLab) project led by UniZH aims to setup and provide a flexible and highly re-usable distributed computing and storage testbed in support of joint research activities across multiple EMANICS partner sites. The testbed, based on the PlanetLab software infrastructure, has nodes located at UniZH, INRIA, UT, UPC, IUB, UniBwM and LMU.

2. The evolution of the Voice over IP Testbed (SIP4EMANICS) project led by INRIA is a continuation of the VoIP testbed project from the first phase of the project. Work items within SIP4EMANICS are distributed VoIP technologies, distributed storage of VoIP account data, call pattern anomaly detection, and call trace data collection. The project has been collaborated by INRIA, UPI, UniZH and UniBwM.

3. The network trace collection and trace maintenance (COLLECT) project led by UT is a continuation of the TRACE project and aims at funding network trace data collection activities. Several partners including UT, IUB, PSNC, UniZH, LMU and UPI are collecting additional traces and developing database schema with the goal to make aggregated information available to interested researchers.

This deliverable reports the achievement of the three projects for the first nine-month period of the second phase. The partners of the work package have met in January 2008 in Barcelona to examine the progress and to identify any standing issues of these projects. The EMANICSLab project describes the setup of a testbed for distributed computing and storage which allows partners to carry out joint research activities. Next tasks include enlarging the testbed within EMANICS partners in terms of resource and usage, and collaborating the testbed with other infrastructures outside EMANICS. The SIP4EMANICS project enriches the VoIP testbed built in the first phase with several features and establishes a P2PSIP overlay on the EMANICSLab testbed. The incentive of exploiting this infrastructure still requires improvement. The COLLECT project continues collecting traces from various networks at partner sites. Further tasks focus on making traces available to EMANICS partners. In addition, these projects are nicely cooperated by more than four partners that have shown a great deal of collaboration.

2 Introduction

The second phase of the EMANICS project lasts eighteen months starting from June 2007 to December 2008. In this phase, the work package remains focusing on three objectives: the establishment of collaboration environments, the development of supporting tools, and the creation and maintenance of trace repositories for research and educational purposes. Nevertheless, the demand of projects are more concerned with collaboration activities, and infrastructure and resource exploitations within EMANICS partners. The work package contains two deliverables in March 2007 and December 2008. An open call for the first nine-month period has resulted in three selected projects:

- The EMANICS Distributed Computing and Storage Testbed (EMANICSLab) project integrates seven partners: UniZH, INRIA, UT, UPC, IUB, UniBwM, LMU and UniS. This project aims to setup and provide a flexible and highly re-usable distributed computing and storage testbed in support of joint research activities across multiple EMANICS partner sites. The initial testbed setup of EMANICSLab shall be based on MyPLC. MyPLC is a complete installation of PlanetLab Central (PLC), the back-end management infrastructure of PlanetLab. UniZH is leading this activity and has already setup PLC and will coordinate this activity. The other partners receiving funding will connect local nodes to EMANICSLab. UniS participates as a user of the infrastructure.

- The Evolution of the Voice over IP Testbed (SIP4EMANICS) project integrates four partners: INRIA, UPI, UniZH and UniBwM. While the initial VoIP testbed has been set up in the first phase, the objectives for the second phase are: (1) to render the system more flexible for experimentation purposes, (2) to extend the integrated services and administration capacities, (3) to gain realistic usage patterns for research purposes, (4) to enable a distributed storage of VoIP accounts, and (5) to extend the existing testbed with a P2P based SIP overlay network. INRIA is leading this project and the other partners contribute on the tasks specified in the proposal.

- The Network Trace Collection and Trace Maintenance (COLLECT) project integrates six partners: UT, IUB, PSNC, UniZH, LMU and UPI. This project aims at funding network trace data collection activities such as the collection of netflow data sets, the collection of full packet traces, or the collection of network management traffic traces. The project is lead by UT. All involved partners committed to collect additional traces and to make them available to partners, subject to the constraints imposed by the network operators.

The three projects stimulate many collaboration activities in other work packages. Several partners take advantage of the EMANICSLab testbed to do joint research activities. SIP4EMANICS provides a new VoIP testbed with several enhanced features including call trace data collection that might foster joint research activities among trace collection researchers. It also deploys a P2PSIP network on the top of EMANICSLab. COLLECT exploits EMANICSLab to store trace data and make them available to partners. A high number of projects built on P2P technology carry out experiments on EMANICSLab. These

projects have succeeded in building up collaborative environments and further inspiring partners to start cooperative research work.

The rest of the deliverables is structured as follows. Section 3 documents the construction of the EMANICSLab testbed, focusing on the configuration, usage and monitoring, of the testbed. Section 4 discusses a SIP4EMANICS testbed based on a VoIP testbed with several improved features. It includes the setup of a P2PSIP overlay on the EMANICSLab testbed. The management trace collection is described in Section 5. Section 6 reports collaboration issues in the work package before the deliverable concludes in Section 7.

3 EMANICSLab

Up to now research activities within EMANICS were running each on their own dedicated testbed infrastructure. The maintenance of such dedicated testbeds - typically spanning multiple EMANICS partner sites - required not only major administration effort, but would also result in a waste of resources if those dedicated testbeds are only used rarely.

Thus, the aim of EMANICSLab was to setup and provide a flexible and highly re-usable distributed computing and storage testbed in support of joint research activities across multiple EMANICS partner sites. Such activities include but are not limited to, distributed trace repositories, distributed flow collection and analysis systems, distributed intrusion detection systems, P2P management architectures, and P2P and SIP integration.

Ongoing and planned research activities in WP7,8,9 are covering several of the above research topics. EMANICSLab provides a single testbed infrastructure which can be used by multiple activities in parallel. Each activity has its own context of virtual servers with root permissions. Thus, they are isolated from each other and give the research activity leader the full flexibility to install new software or perform configuration changes as needed.

EMANICSLab also allows the deployment of distributed services that have reached a stable state to go into production, *e.g.*, a trace repository or a large-scale storage system that can be used for activities in WP1,3,4,5,6.

The initial testbed setup of EMANICSLab is based on MyPLC [1]. MyPLC is a complete installation of PlanetLab Central (PLC), the backend management infrastructure of PlanetLab [2]. The installation of a dedicated research testbed for EMANICS is to be preferred over the use of PlanetLab itself for the following reasons: First, resources in PlanetLab are quite limited, *e.g.*, the standard disk quota is only 5GB per user on each node, which is easily exceeded by certain research activities like distributed flow collection. Second, the testbed configuration and control is done centrally by the PlanetLab administrators which are not always able to accommodate the needs users might have. Moreover, this can be a problem for systems like trace repositories which typically require strong trust relationships and access protection mechanisms.

The setup of a dedicated EMANICSLab enables the flexible allocation of resources to research activities within EMANICS and ensures that the control of the testbed stays within the NoE and access to it can be restricted if necessary. Moreover, extensions or changes to the testbed, *e.g.*, towards the use of a different virtualization platform can be done.

The main result from this project is a fully functional, highly re-usable testbed infrastructure across multiple EMANICS partner sites, supporting several research activities in parallel in a very flexible manner. In the following sections, an overview on the research activities running on top of EMANICSLab is given, and statistics about the usage of resources within EMANICSLab are provided. Additionally, the research activities and basic testbed services running on top of EMANICSLab are briefly described.

3.1 Testbed Setup and Configuration

The EMANICSLab PLC node is located at the Department of Informatics of University of Zurich. The node was installed using the MyPLC package provided by Planet-Lab devel-

opers. Besides the installation as described in the installation manual a set of files of the web interface have been changed in order to correct some errors related to authentication of calls and outdated method calls.

A backup script has been written, which takes a daily backup of the Postgres database on EMANICSLab central, including the complete EMANICSLab configuration with all sites, nodes, users, and slices.

The principal goal was to have a basic version of EMANICSLab (based on MyPLC) being established as soon as possible, in order to be able to use it in the different research activities.

After having gained some experience with that basic version, a discussion was started on potential extensions and changes of the basic EMANICSLab setup and configuration. This discussion took place during the WP2 meeting early 2008 in Barcelona and included participants from most EMANICSLab sites and research activities using EMANICSLab. The discussion resulted into a couple of testbed changes which will be proposed for the next phase.

In order to bootstrap EMANICSLab, UniZH has setup an EMANICSLab central management server accessible at [3], which allowed the partners to integrate their nodes into EMANICSLab, add users and slices, etc. like in PlanetLab. A screenshot of the management interface is depicted in Figure 1.

The following were the necessary steps to be undertaken by each EMANICSLab partner:

- At least one person from each EMANICSLab partner had to create an account and that will be PI and TC. This was achieved by going to https://emanicslab.csg.uzh.ch/ and selecting ”Create an account”. The corresponding site and the roles ”Principal Investigator” and ”Technical Contact” had to be selected.
- Two nodes (x86-based server-class machines) had to be prepared to be integrated into EMANICSLab. (Ideally and if possible, the nodes had to be placed outside the firewall to both isolate them from the campus network and to allow unfiltered access to them. Moreover, the nodes needed to have a public IPv4 address. More information about suitable hardware is provided under http://www.planet-lab.org/doc/guides/tech.
- The option ’Add Node’ had to be selected and the necessary information for each node (IP address, etc.) had to be provided.
- From the node’s page the configuration file had to be downloaded and written to a floppy or USB stick.
- The BootCD image could be downloaded from http://emanicslab.csg.uzh.ch/download /EmanicsLab-BootCD-3.3.iso and was used to burn the boot CD.
- The floppy or USB stick with the configuration file had to be inserted in the machine.
- Finally, the machine had to be booted from CD in order to install the node.

Normally, the nodes were installed with a generic boot CD and a configuration file on a floppy disk or USB stick. However, some nodes could not be installed in this manner.

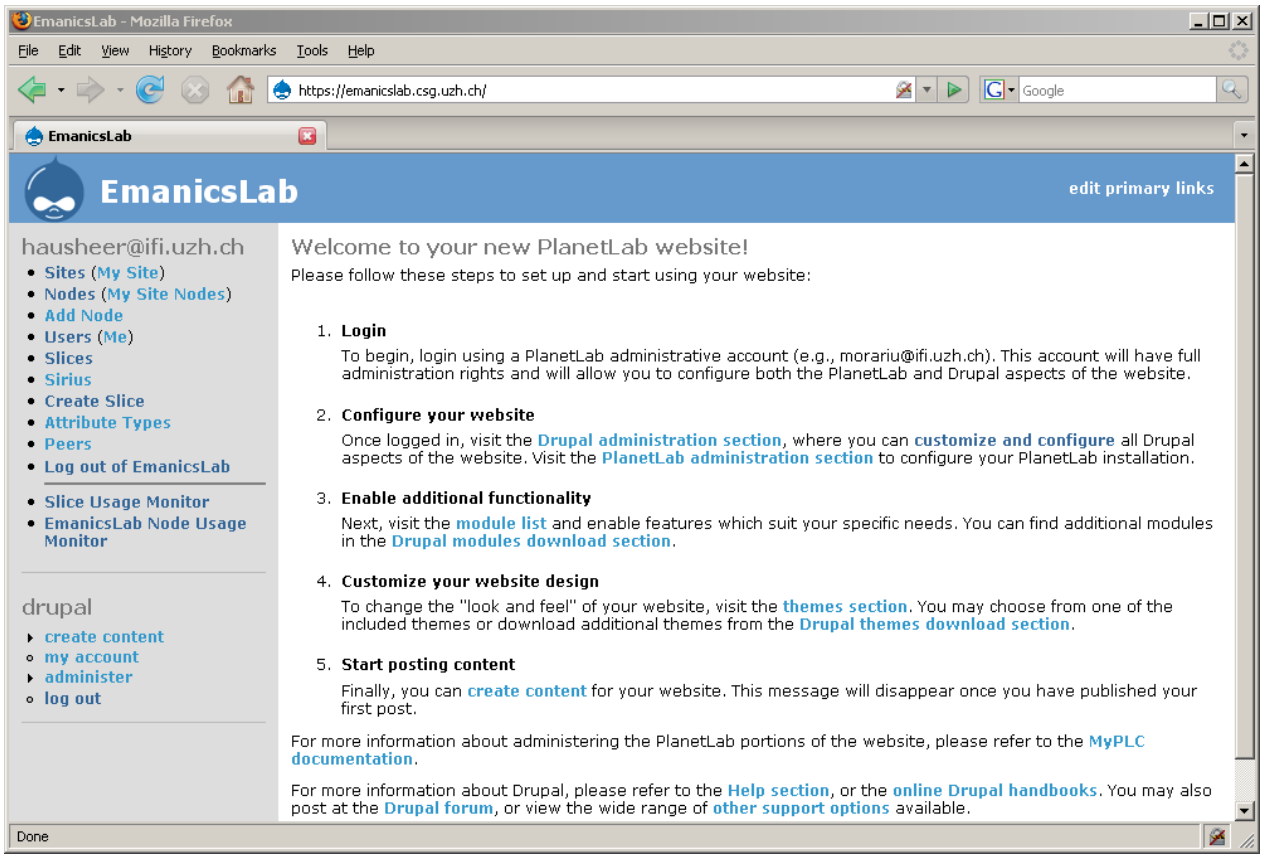


Figure 1: Screenshot of EMANICSLab Website

For these nodes, a custom boot CD had to be created using the build-script included in myPLC:

```
/usr/share/bootcd/build.sh -f /planet.cnf
cp EmanicsLab-BootCD-3.3.iso /var/www/html/download/boot-xxx.iso
rm EmanicsLab-BootCD-3.3*
```

Further steps needed to make use of EMANICSLab:

- Each research activity member needs to create a personal account. The PI for each site is in charge to enable users and assign them to slices. Users need to upload their public key, in order to be able to login to EMANICSLab nodes.
- Research activity members can add as many nodes to their slice as they want. A virtual root environment will be created on every node selected. By default every slice has a certain resource limit per node. These limits can be increased upon request to the EMANICSLab administrators.

Table 1 provides a summary of the key problems that came up during the setup and configuration of EMANICSLab, as well as their solution.

Problem	Solution
Ports are closed from outside the University	Ask network administrators to open ports in the firewall. Ideally, all ports should be open.
Login to nodes as site_admin not permitted	Resolved manually, seems to be a bug. Only used to reboot nodes anyway.
Config file cannot be found	Config file on floppy/USB has to be renamed to plnode.txt. Custom boot CD images provided by UniZH.
Dell Precision 380 does not boot CD	Switched to a Dell Precision 390 instead.
HP DC7700 does have PCI problems	Requires kernel option pci=nommconf
Newer RAID card not supported by the kernel	Switched to a different machine without RAID card
Installation of software does not work	Replace yum config file /etc/yum.conf
Bind to privileged ports not possible	Only unprivileged ports ($>$ 1024) can be used within a slice

Table 1: EMANICSLab Problems and their Solution

For people interested in additional details of EMANICSLab, the EMANICSLab tutorial [4] which was given at the EMANICS WP2 meeting in Barcelona may provide further valuable information. Furthermore, in order to get support, the EMANICSLab administrators can be contacted at emanicslab@ifi.uzh.ch.

3.2 Testbed Overview

An overview on the different sites and nodes in EMANICSLab is provided in Figure 2. At this stage, the EMANICSLab research network includes 8 sites with 14 nodes. The different sites are outlined in Table 2. Every partner site (except UniS which is only a user of the testbed) provides 2 nodes as outlined in Table 3. Furthermore, for every site a principal investigator (pi) and a technical contact (tech) had to be determined. Additionally, UniZH serves as EMANICSLab Administrator (admin). The key users and their roles are outlined in Table 4.

3.3 Testbed Usage

At this stage, 11 research activities are using EMANICSLab in their research (cf. Table 5). There is a slice for every research activity using EMANICSLab. A slice is a set of slivers, i.e. virtual servers on different nodes.

3.3.1 DATTA (uzh_datta)

The key aim of this project is the design and prototypical implementation of a distributed NETFLOW collection platform integrated within EMANICSLab. The flow records collected

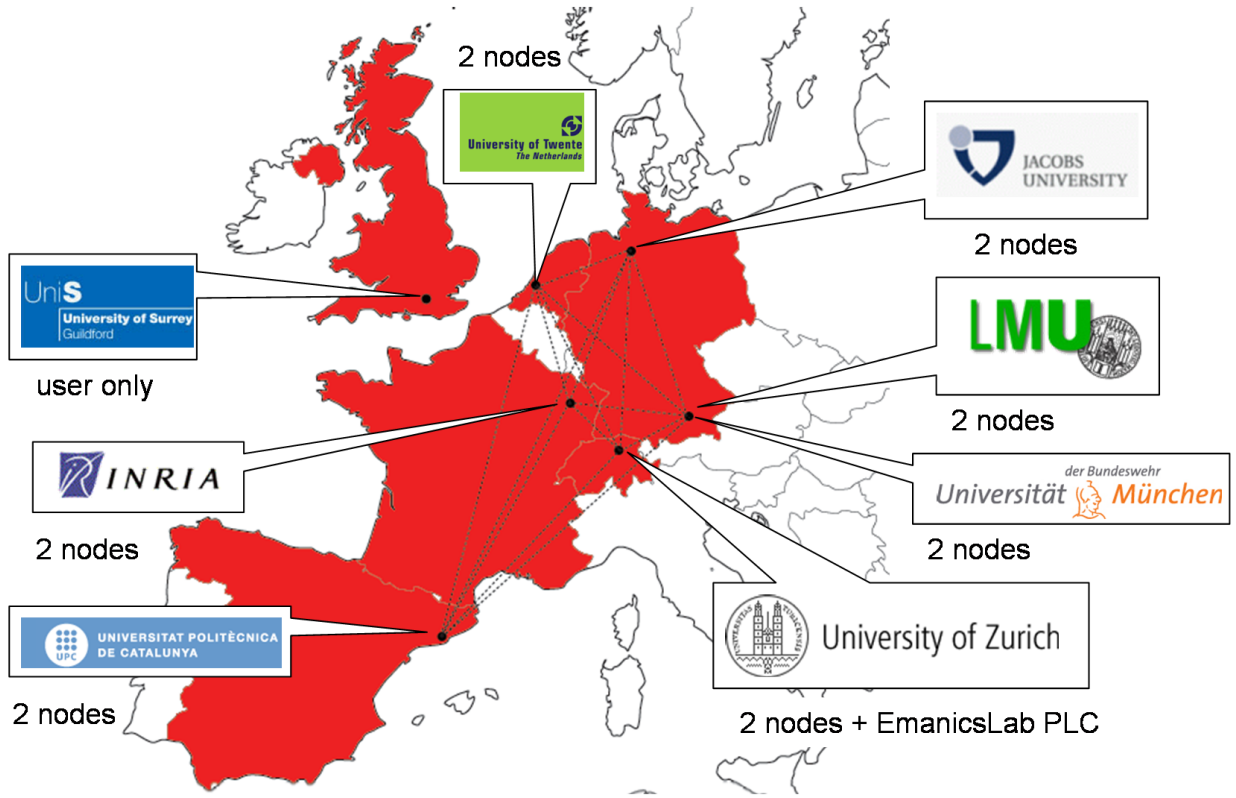


Figure 2: EMANICSLab Research Network

Abbreviated Name	Name	Login Base
EMANICSLab	EMANICSLab Central	pl
INRIA	Institut National de Recherche en Informatique et Automatique	inria
IUB	Jacobs University Bremen	iub
LMU	Ludwig-Maximilian University Munich	lmu
UPC	Universidat Politechnica de Catalunya	upc
UniBw	University of Federal Armed Forces Munich	unibw
UniS	University of Surrey	unis
UT	University of Twente	ut
UZH	University of Zurich	uzh

Table 2: EMANICSLab Sites

by each partner are stored locally on a separate host within the premise of the partner. The platform offers a client interface for requesting flow records from any of the participating partners whereas the EMANICSLab nodes act as brokers between clients and storage repositories and additionally have the task to control access. Moreover, the platform offers an API that allows developers to build their own applications on top of the provided library.

Site	Hostname	Model
INRIA	host1-plb.loria.fr	Dell Precision 360, Pentium 4, 3.0 GHz, 2 GB RAM, 750 GB HDD
INRIA	host2-plb.loria.fr	Dell Precision 390, Core 2 X6800, 2.93 GHz, 3 GB RAM, 500+250 GB HDD
IUB	emanicslab1.eecs.jacobs-university.de	Dell OptiPlex GX620, Pentium D, 2.8 GHz, 1 GB RAM, 80 GB HDD
IUB	emanicslab2.eecs.jacobs-university.de	Dell OptiPlex GX620, Pentium D, 2.8 GHz, 1 GB RAM, 80 GB HDD
LMU	emanicslab1.lab.ifi.lmu.de	Fujitsu Siemens Scenic, Pentium 4, 3.0 GHz, 1 GB RAM, 400+400 GB HDD
LMU	emanicslab2.lab.ifi.lmu.de	HP DC7700, Core 2 6400, 2.13 GHz, 3.6 GB RAM, 80+500+500 GB HDD
UPC	muro.upc.es	Custom Model, Core 2 6400, 2.13 GHz, 1 GB RAM, 250 GB HDD
UPC	moscu.upc.es	Custom Model, Athlon XP 1600+, 1.4 GHz, 1 GB RAM, 200 GB HDD
UniBw	emanicslab1.informatik.unibw-muenchen.de	Dell PowerEdge 2850, Xeon, 3.0 GHz, 2 GB RAM, 150 GB HDD
UniBw	emanicslab2.informatik.unibw-muenchen.de	Dell PowerEdge 2850, Xeon, 3.0 GHz, 2 GB RAM, 150 GB HDD
UT	emanicslab1.ewi.utwente.nl	Dell PowerEdge 860, Dual Core Xeon 3070, 2.66 GHz, 4 GB RAM, 2 TB HDD
UT	emanicslab2.ewi.utwente.nl	Dell PowerEdge 860, Dual Core Xeon 3070, 2.66 GHz, 4 GB RAM, 2 TB HDD
UZH	emanicslab1.csg.uzh.ch	Dell PowerEdge 850, Pentium 4, 3.6 GHz, 1 GB RAM, 500 GB HDD
UZH	emanicslab2.csg.uzh.ch	Dell PowerEdge 850, Pentium 4, 3.6 GHz, 1 GB RAM, 500 GB HDD

Table 3: EMANICSLab Nodes

3.3.2 IPLoc (uzh_iploc)

The IPLoc slice is used for a distributed evaluation test within the frame of the University of Zurich diploma thesis ”Mechanisms for Mapping End Systems in the Internet to Geographic Location Information”. IPLoc, which constitutes one out of two prototype implementations developed and evaluated in this work, was deployed on the slice’s nodes and run in parallel for the purpose of determining potential variations in reply delays when running IPLoc in a distributed manner.

Site	Firstname	Lastname	Email	Roles
INRIA	Emmanuel	Nataf	nataf@loria.fr	pi, tech
IUB	Juergen	Schoenwaelder	j.schoenwaelder@jacobs-university.de	pi, tech
LMU	Feng	Liu	liufeng@mnm-team.org	pi, tech
UPC	Pau	Valles	pvalles@nmg.upc.edu	pi, tech
UniBw	Frank	Eyermann	frank.eyermann@unibw.de	pi, tech
UniS	Stylianos	Georgoulas	s.georgoulas@surrey.ac.uk	pi, tech
UT	Ramin	Sadre	sadrer@ewi.utwente.nl	pi, tech
UZH	Cristian	Morariu	morariu@ifi.uzh.ch	admin, pi, tech
UZH	David	Hausheer	hausheer@ifi.uzh.ch	admin, pi, tech
UZH	Thomas	Bocek	bocek@ifi.uzh.ch	pi, tech

Table 4: EMANICSLab Users

3.3.3 P2PFastSS (uzh_fastss)

This slice is used for experiments with P2P Fast Similarity Search (P2PFastSS). P2PFastSS is a fast similarity search algorithm for structured P2P systems. The algorithm allows to search for similar keys in any distributed hash table (DHT) using the edit distance metric. Thus, P2PFastSS is independent of the underlying P2P routing algorithm. P2PFastSS is suitable for similarity search in large-scale network infrastructures, such as service description matching in service discovery or searching for similar documents in P2P storage networks.

3.3.4 P2P-SIP (inria_p2psip)

This slice is used for the deployment of a P2P-SIP network on top of EMANICSLab. The slice creates a P2P overlay network using the open source Bamboo DHT [5] and has a P2P-SIP implementation [6] that runs on top of the overlay network. The Bamboo DHT not only creates a P2P overlay network for the P2P-SIP network, but can also be used as an OPEN DHT service [7] for other P2P applications. The P2P-SIP implementation can be used by other partners who wish to use or do research on P2P-SIP communication. A detailed explanation of the P2P-SIP integration is provided in the SIP4EMANICS activity.

Slice Name	Leading Partner	Other Partners	Users
uzh_datta	UniZH	UT, PSNC, LMU	Cristian Morariu, Nicolas Baumgardt, Liu Feng
uzh_iploc	UniZH	-	Martin Waldburger, Stefan Boesch
uzh_fastss	UniZH	-	Fabio Hecht, Dalibor Peric, Thomas Bocek
inria_p2psip	INRIA	UniZH, UPI, UniBwM	Balamurugan Karpagavinayagam
inria_p2prevocation	INRIA	-	Thibault Cholez
uzh_voip	UniZH	INRIA, UPI, UniBwM	Stefan Huber, Gregor Schaffrath, Mark Furrer
iub_buglook	IUB	LMU, UniS	Juergen Schoenwaelder, Ha Manh Tran, Georgi Chulkov
upc_sblomars	UPC	-	Pau Valles
inria_jump	INRIA	-	Emmanuel Nataf
ut_snid	UT	UniZH, PSNC, IUB, UniBwM	Ramin Sadre
unibw_asam	UniBwM	UniZH, UniS	Frank Eyermann

Table 5: EMANICSLab Slices

3.3.5 P2P Revocation (inria_p2prevocation)

The P2P revocation slice is used to run a modified aMule client connected to the widely deployed structured P2P network KAD (Kademlia). The newly designed revocation mechanism distributes information about who has to be revoked. This revocation information can only be managed by the modified client and needs to be replicated on several peers. The slice has been used to study how the number of peers storing the information affects the mechanism's performance in the real P2P network. EMANICSLab gives the possibility to run a client on different nodes and to connect it to KAD over a direct Internet connection.

3.3.6 VoIP (uzh_voip)

This slice is allocated for SIP4EMANICS related experiments. It serves the following four purposes: (1) Feasibility evaluation to run actual asterisk instances inside of EMANICSLab, (2) prototypical implementation of an experimental VoIP environment as proof of concept of the defined SIP4EMANICS VoIP architecture design, (3) evaluation of potential issues with respect to specific asterisk components in the EMANICSLab environment, and (4) field for initial experimentation and sandbox for potential SIP4EMANICS research users to allow familiarization with the environment and concepts before the actual project draft and start.

3.3.7 Buglook (iub_buglook)

This slice is used for the testing and evaluation of buglook, a web crawler for bug tracking systems. The buglook system is indexing bug trackers and storing information in a unified bug tracking data model. Feature vectors and semantic vectors are computed in order to search in the extracted data base. The data will be accessible through a simple web frontend and there will be an interface to a distributed case-based reasoning system. The slice is therefore linked to research in WP9 of the EMANICS project.

3.3.8 SBLOMARS (upc_sblomars)

The purpose of this slice is to deploy SBLOMARS, a distributed network monitoring system to track the usage of network node resources as well as end to end network transmission parameters. The system is installed in each network node and after booting it instantiates as many monitoring agents as needed to monitor the different resources existing in the node (one agent per resource). All these agents run as independent software threats. Data in different time windows is stored in the local node, ready to be exported to any other system. When used in cooperation with a scheduling system, the data collected by the monitoring agents will be exported to the scheduler. Nevertheless, scenarios with a P2P data exchange mechanism could also be considered. Initially conceived for large-scale grids, the system has been adapted to be deployed in EMANICSLab. Potential use of this monitoring system is to help the Lab administrator in his role but also guide users when they plan to deploy their experiments. In its current version, the system is providing per node memory and CPU usage only. The intention is to make the appropriate adaptation of the system in order to be able to monitor at a per slice basis. Data is provided through a web interface that shows the nodes usage snapshots updated every minute.

3.3.9 JUMP (inria_jump)

This slice is used to deploy the Java implementation of JXTA in order to run a peer-to-peer application on several nodes of the slice. All these peers have a management interface implemented with JMX, which is used as a management plane in order to monitor and eventually control community services. The management plane is capable of self-organization by a manager election mechanism running within the peer-to-peer network.

3.3.10 SNID (ut_snid)

Although sophisticated algorithm for network intrusion detection yield impressive results under laboratory conditions, they often do not scale to real-life networks due to the sheer amount of data transferred in the latter. The goal of the SNID research activity is to design sampling-based algorithms for the detection of well-known intrusion patterns in network traces. In this context, UT has collected NETFLOW traffic traces that are used for the evaluation of existing and the development of new intrusion detection algorithms. Since the traces stem from high-speed networks, their sizes pose a challenge to the evaluation:

a two-day trace comprises around one billion records. In order to abstract from the underlying storage technique, the traces are stored in a relational SQL database. Instances of the database run in this slice.

3.3.11 ASAM (unibw_asam)

This slice is used for experiments within the ASAM activity. The objective of ASAM (Auditing of SLOs Across Multiple Provider Domains) is to develop an architecture and a new protocol for metering and auditing of network performance across multiple, co-operating network providers. In this context the auditing of Service Level Agreements (SLA) defines the process of monitoring whether a service provider delivers agreed upon service levels or not. While frameworks exist to monitor applicationlevel SLAs, the end-to-end monitoring of IP-carrying SLAs, especially in a multi-domain environment like the Internet, is still an open issue. Measurements from within EMANICSLab will be used to parametrize ASAM simulation scenarios.

3.4 Testbed Monitoring

The PlanetLab web pages [2] provides a list of monitoring tools for PlanetLab. For EMANICSLab the following three monitoring applications have been considered: PlanetFlow, CoMon, and Ganglia. These three monitoring applications seem to be actively maintained. While every PlanetLab node has PlanetFlow built in, CoMon and Ganglia has to be installed additionally.

3.4.1 PlanetFlow

This monitoring software provides an overview over network flows. Each slice can be monitored separately. The following measurements are collected: number of flows, number of packets, flow size in megabytes, and source and destination IPs. The data can be browsed either by slice or by source. In addition to that, these measurements are archived and can be accessed online for 30 days. These logs are transferred to the EMANICSLab central and there the logs are stored for 8 weeks. In order to access older logs, the EMANICSLab central archives these logs. Access to those log files are granted for all EMANICS members. The statistical data for, *e.g.*, the UniZH EMANICSLab node 1 can be found here: http://emanicslab1.csg.uzh.ch:1080/.

The major advantage of PlanetFlow is that it is already installed. The drawback is that PlanetFlow does only provide flow information. Further information, such as CPU usage is not available in PlanetFlow.

3.4.2 CoMon

CoMon is a service offered by Princeton and it runs on top of PlanetLab. This service shows the performance of nodes with respect to CPU speed, load on the system, swap

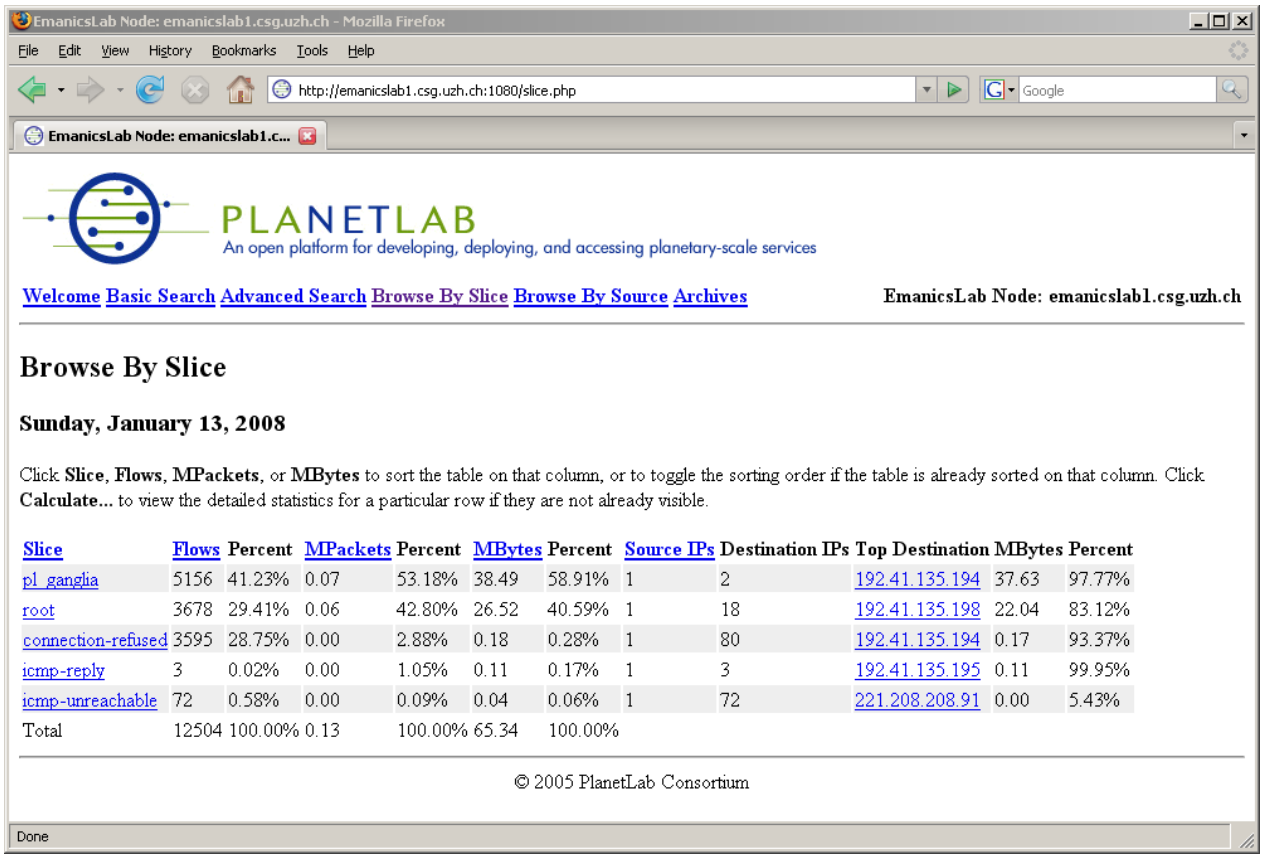


Figure 3: Screenshot of PlanetFlow in EMANICSLab

and disk space, and bandwidth. It can identify nodes and slices with resource problems, such as a high load, or low disk space.

CoMon can be accessed through a web browser and almost every data can be converted to a graph. In addition, CoMon allows user defined queries for fetching statistical data. For example *select='resptime* $>$ *0'* shows all nodes that are alive, or *select='resptime* $>$ *0 && gbfree* $<$ *5'* shows all nodes that have less than 5 GB free disk space. More information can be found here: http://comon.cs.princeton.edu/.

The major advantage of CoMon is that it is highly customizable. CoMon can provide statistical data for other tool or applications. The major drawback is that it cannot be installed on EMANICSLab, as the source code is not available at the moment.

3.4.3 Ganglia

Ganglia [8] is a monitoring application for cluster systems and Grid networks. It uses well known technologies such as XML for data representation and RRDtool for data storage and visualization. Ganglia has been ported to many operating systems and it does also run on PlanetLab. Many organizations are using Ganglia, *e.g.*, CERN, MIT, Berkeley, Reuters, Wikipedia, and Microsoft.

Ganglia has two daemons and a web frontend. The monitoring daemon, which runs on

every node, checks for changes in the state of a node and answers queries for the current state. The meta daemon collects data from the monitoring daemon. This daemon can be built in a tree structure for scalability reasons, however, due to the relative small size of EMANICSLab, the meta daemon runs on EMANICSLab central only. The meta daemon polls periodically each monitoring daemon for its state. The data is then passed to a round-robin database. The frontend is a PHP application which displays the data from the round-robin database. The data is displayed in realtime and shows the CPU usage, memory usage, disk usage, and network usage. As the frontend updates the data and builds an XML tree at every request, the frontend should run on a powerful machine.

3.4.4 Ganglia on EMANICSLab

Ganglia has been installed on EMANICSLab to measure the usage. A detailed description of the installation follows.

Each node has to install the monitoring daemon, which is available as an RPM (Red Hat Package Manager) on the Ganglia project site. The command: *rpm -Uhv ganglia-gmond-3.0.3-1.fc4.i386.rpm* installs and starts the monitoring daemon. The configuration file for the monitoring daemon has been adapted. The property name and value “name = unspecified” has been changed to “name = EmanicsLab”.

On the server side, the meta data daemon and a web frontend have to be installed using the following commands *rpm -Uhv ganglia-gmetad-3.0.3-1.fc4.i386.rpm* and *rpm -Uhv ganglia-web-3.0.3-1.noarch.rpm*. Every EMANICSLab node (*cf.* Table 3) has to be listed in the gmetad.conf file.

It may be necessary to install *rrdtools, apache2, and php* if not present on the server. Figure 4 shows EMANICSLab graphs with CPU, load, memory, and network usage. The site can be found at http://emanicslab.csg.uzh.ch/ganglia/.

Figures 5, 6, and 7 show a 1 minute load average, the number of processors and the running threads. These graphs show the resource usage for one month.

Figure 5 shows the total amount of resources used in EMANICSLab. In week 9, there has been a peak of running processes in EMANICSLab, and in week 8, a peak is visible for the load average.

Figure 6 shows the amount of resources used for the UT nodes. This clearly shows that experiments on these nodes lead to the peak of the load average in week 8. Other nodes do not show this behaviour in week 8. This graph also shows the peak in week 9.

Figure 7 shows the amount of resources used for the UniBW nodes. In this figure, the peak in week 8 is not present, but there is a peak in week 9. Thus, it is very likely that the peak of number of running processes in week 9 was due to a global test or simulation on EMANICSLab and the peak in week 8 is likely to be a test or simulation on a local node.

The load average for the UniBW nodes is around 1, which means that these nodes are regularly used for testing or simulations. The UT nodes show a smaller load average, but with a peak for the load average of up to 14.

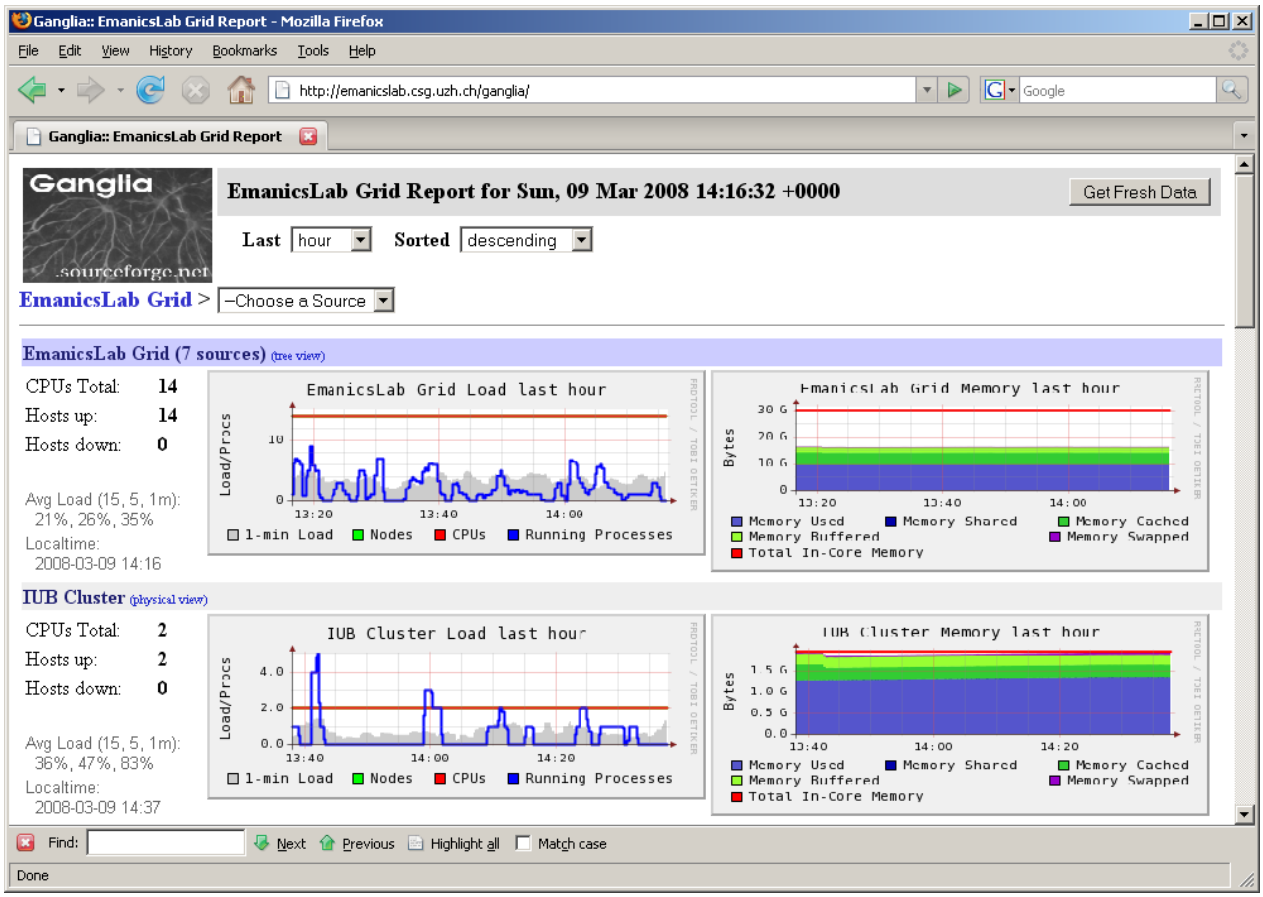


Figure 4: Screenshot of Ganglia on EMANICSLab

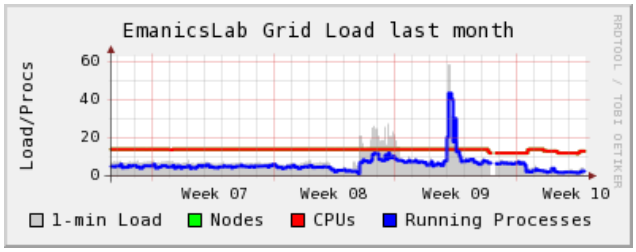


Figure 5: EMANICSLab Grid (7 sources) load, CPUs, and running processes

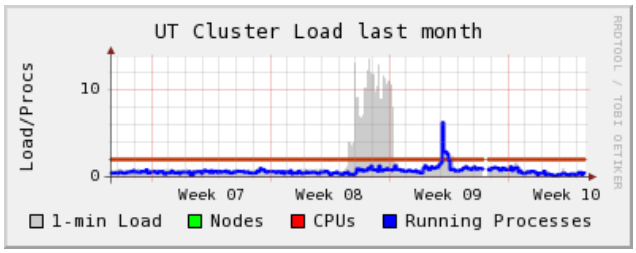


Figure 6: UT Cluster load, CPUs, and running processes

3.4.5 ElabMoni

ElabMoni [9] is a monitoring tool for EMANICSLab developed by UniZH. It monitors the resource usage of different slices. ElabMoni consists of an agent running on all EMAN-

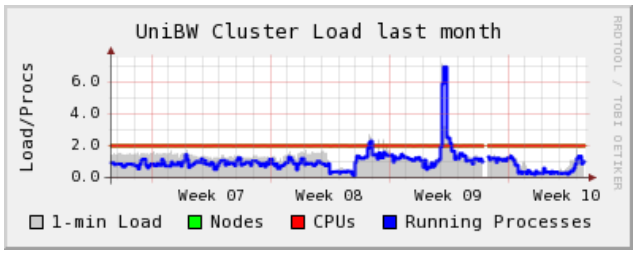


Figure 7: UniBW Cluster load, CPUs, and running processes

Host split for slice uzh_fastss for the last 60 seconds:

Hostname	Total CPUs	Total MEM
emanicslab1.informatik.unibw-muenchen.de	0.0233	10.45 MB
emanicslab2.lab.ifi.lmu.de	0.0133	10.61 MB
emanicslab2.ewi.utwente.nl	0.0125	11.87 MB
emanicslab2.informatik.unibw-muenchen.de	0.0116	69.19 MB
moscu.upc.es	0.0091	81.97 MB
emanicslab1.lab.ifi.lmu.de	0.0066	89.99 MB
emanicslab1.eecs.jacobs-university.de	0.0066	10.38 MB
emanicslab2.eecs.jacobs-university.de	0.0041	11.46 MB
host1-plb.loria.fr	0.0000	0.00 MB
host2-plb.loria.fr	0.0000	0.00 MB
emanicslab2.csg.uzh.ch	0.0000	11.33 MB

Figure 8: Detailed usage of a single slice

Host usage of host emanicslab1.lab.ifi.lmu.de for the last 60 seconds:

Slice Name	Total CPUs	Total MEM
ntpia_manuslice	0.0000	0.00 MB
pl_ganglia	0.0000	2.87 MB
pl_netflow	0.0066	54.33 MB
rootetflow	0.0000	0.00 MB
rpct	0.0000	0.00 MB
rpcuser	0.0000	0.00 MB
smmsper	0.0000	0.00 MB
upc_sblomars	0.1111	38.03 MB
user blomars	0.0000	0.00 MB
uzh_fastss	0.0066	89.99 MB

Figure 9: Detailed usage for an EMANICSLab node

ICSLab hosts and a monitoring station that collects data from the monitored hosts and presents them via a web page. ElabMoni agent is installed automatically on each EMANICSLab node during node installation phase. Updates are also automatically installed during the node update process (which runs once a day on every EMANICSLab node). The monitoring station can show how much resources (CPU and memory) a slice uses on a particular node or it can show aggregated values over all nodes on which the slice is instantiated.

ElabMoni statistics can be seen at http://emanicslab.csg.uzh.ch/elabmoni/. Figures 8 and 9 show different statistics that can be compiled by ElabMoni. In Figure 8 the detailed usage of a single slice can be observed. ElabMoni allows to see the CPU usage and memory usage of a single slice on all EMANICSLab nodes on which the slice has a sliver. Figure 9 shows the detailed statistics of a single EMANICSLab node.

4 SIP4EMANICS

This project contains several activities. The sub-section 4.1 introduces the new architecture of a SIP4EMANICS (S4E) testbed and clarifies the terminology with respect to S4E participants and S4E users. An overview of participation scenarios among which S4E participants can choose is given in the sub-section 4.2. Each scenario contains a description on how to configure the distributed installations and how to integrate possibly existing remote user accounts into the SIP4EMANICS testbed. The sub-section 4.3 defines the stable production environment, related components and management interfaces for distributed management. The sub-section 4.5 explains how the related components defined in sub-section 4.3 are used to collect usage patterns and to make them available. The testing environment is described in the sub-section 4.4, where research prototypes can be integrated into the environment. Particularly, this environment re-uses the EMANICSLab installation and is designed with only minimal integration into the stable part in order not to put the production character of the installation at risk. The sub-section 4.7 presents the architecture and integration of the P2PSIP in the EMANICS VoIP testbed before the last activity illustrates the Nagios monitoring architecture for the P2PSIP in the sub-section 4.8.

4.1 Architecture

The new S4E architecture has been designed with regard to flexibility. Its core consists of a stable production system which features an interface to experimental functionality of VoIP research projects within EMANICS. It also provides functionality for collecting Call Detail Records (CDRs) which fosters EMANICS partners to do research on VoIP trace data.

This architecture considers institutions participating in S4E as *S4E participants* and S4E VoIP users as *S4E user*. Figure 10 shows the overview of the architecture with the core system labelled by *s4es*, to which VoIP labs from S4E participants connect for establishing a stable production system. The context descriptions are given in pseudo code similar to the syntax used in Asterisk's extension.conf [10] configuration file.

4.2 Participation

Three different participation scenarios have been proposed for EMANICS members to participate in S4E. Each S4E participant relies on his resources and needs to choose the most suitable scenario.

4.2.1 Integration of a Local Production System

If S4E participants already have a local productive VoIP installation with a personal user management in place, they can choose to integrate their system to S4E. An example for this scenario is given in Figure 10, instantiated by *Partner1*.

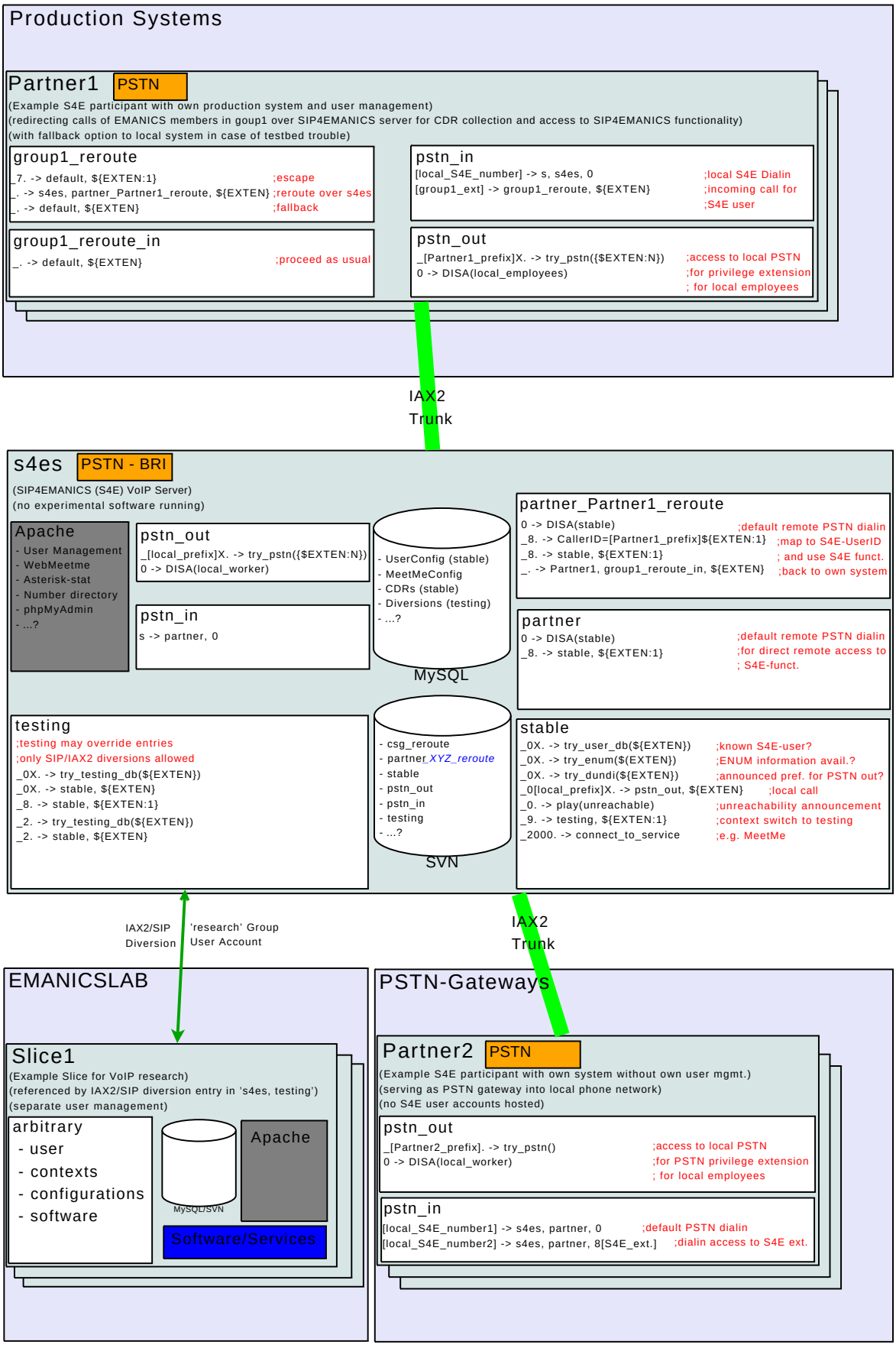


Figure 10: S4E Architecture

In this scenario, EMANICS members at the S4E participant's site are assigned a special context *group1_reroute*, which attempts, by default, to route calls over the corresponding *partner_Partner1_reroute* context on *s4es*. These calls are sent back to context *group1_reroute_in* on *Partner1* by default and thereafter treated as normal calls on the local system without further using S4E functionality. These calls also facilitate the collection of realistic CDRs.

If S4E users playing the call desire to access the S4E functionality, they can dial *8*, followed by a number matching an entry in the *stable* context. In order to ensure the seamless functioning of *Partner1* even in the event of a failure of *s4es*, a fallback entry and another local prefix (*7* in the example) overriding the reroute is placed into *group1_reroute*.

S4E participants choosing this participation scenario are required to create additional VoIP accounts for their users on *s4es* in order to ensure their addressability inside S4E and to implement a CallerID mapping into their respective *partner_Partner1_reroute* for consistency.

Incoming calls for S4E local users (handled in *pstn_in*) are routed over *group1_reroute*, *partner_Partner1_reroute* and thereafter handled again by *Partner1*. CDRs on *s4es* are created for incoming calls as well.

Due to a possible conflict with local dialing schemes, the prefixes proposed in *group1_reroute* and *partner_Partner1_reroute* can be replaced by arbitrary digits by each partner when the functionality is provided. This issue only affects a partner's S4E users.

4.2.2 Provision of a PSTN Gateway

If S4E participants have a local VoIP installation featuring an interface to the local PSTN network and either don't have a local user management in place or wish to keep productive use of this system disconnected from S4E, they can choose to offer PSTN gateway functionality to the local PSTN network to S4E. In Figure 10, this scenario is instantiated by *Partner2*.

S4E participants create a *pstn_out* context, in which they allow S4E users to place calls to the local PSTN network via an IAX2 trunk to *s4es*. A local extension ('0' in the example) is connected to an authenticating DISA [11] application to allow users full dialout capacities even when using their S4E accounts.

S4E participants allocate at least one PSTN extension in *pstn_in* to S4E, which is mapped onto the DISA application in the *partner* context on *s4es* to allow access to S4E over PSTN. They may assign additional numbers to direct access of S4E features by mapping them to '8', followed by the respective extension in the *stable* context on *s4es*.

The prefixes, to which S4E participants offer gateway services, are published via DISA. S4E user accounts in this scenario are hosted on *s4es* and assigned to either the *stable* or *testing* context. In addition, digits *0* and *8* in *partner* are fixed because they don't interfere with the behaviour of other systems but S4E.

4.2.3 Participation without a Local VoIP Installation

If S4E participants don't have a local VoIP installation which they wish to share with other S4E participants, they can choose to participate S4E by registering their members with S4E user accounts, which are hosted on *s4es* in Figure 10 and placed in either the *stable* or *testing* context.

4.3 Stable Production Environment

In order to provide a stable production environment, experimental software is not allowed to run on *s4es* shown in Figure 10. While a *testing* context is defined on *s4es*, it contains merely diversion information to access the testing functionality running on other VoIP labs, and stable functionality is not allowed to build upon the *testing* context.

4.3.1 Stable Context

As S4E users are geographically distributed, the dialout extensions in the *stable* context are kept independent from any locality. Users and PSTN numbers can be called by prepending a 9 to a phone number in the format of

[CountryCode][RegionCode][PhoneNumber]

(*e.g.*, 41446350890). Testing functionality can be accessed by prepending a 9 and thereby explicitly triggering a context switch to the *testing* context. Stable services can be accessed by dialling 2000, followed by a four-digit service extension (*e.g.*, 20001000 may be assigned to the MeetMe [12] conference application).

Dialling of the prefix *0* results in a series of consecutive steps:

1. First the local user and diversion database is queried. If an entry is found, the call proceeds as specified in the database.
2. If no local entry is found in the database, ENUM [13] records specifying SIP or IAX2 destinations are considered. If an acceptable ENUM record has been found, the call proceeds as indicated by the record.
3. If no ENUM entry is found, published DUNDI [14] information of partners is considered. If a S4E participant offering connectivity to the destination is found, the call is forwarded to the respective S4E participant.
4. If the call is local to *s4es*, the local PSTN interface is used.
5. If all previous steps fail, the caller is signaled the failure of the placement of his call.

In order to avoid collisions between S4E participants and to provide a consistent environment, user identifications are based on their office phone numbers, possibly extended by a one-digit counter in case multiple S4E users share one office phone [1].

While the stable context is mostly implemented in Asterisk's extension.conf [10] or included therein, users and specific service configurations are stored in a local SQL database for facilitating dynamic reconfiguration via PHP scripts running on a local webserver. This part makes extensive use of the Asterisk RealTime Architecture [15].

If partners prefer a localized version of the *stable* and *testing* contexts, individual replication can be considered at the later stage.

4.3.2 Databases

The stable environment relies on two databases. A MySQL [16] server hosts dynamic configuration parts, like user configurations, user defined diversions, MeetMe conference configuration, CDRs and testing diversions. A SVN [17] repository stores critical configuration files and context descriptions, which thereby can be quickly restored after a possible erroneous change, or which can be distributed to other sites at the future stage.

4.3.3 Management Interface

A web server running PHP scripts provides a management interface which supports for distributed management of S4E participants. User Management is available at three different access levels:

1. S4E administrators can define groups and assign these groups to S4E participant administrators.
2. S4E participant administrators can add/delete/modify S4E user accounts in their respective namespace given by their institution's phone prefix. (*e.g.*, 414463 for UniZH)
3. S4E users are able to configure their respective account details as well as diversions for their extensions.

Figures 11, 12 and 13 depicts different view examples. The WebMeetMe [18] interface is installed for dynamic conference management.

4.4 Testing Environment

The S4E architecture in Figure 10 features a *testing* context to experimental use of S4E. Experimental software is not run on *s4es* and developers do not receive low level access to *s4es*. However, the context is structured in a way to query diversion information dynamically from the MySQL [16] database via the Asterisk RealTime Architecture [15].

[1] This digit needs to be removed in the *pstn_out* context of the local VoIP installation of the respective S4E participant, if applicable

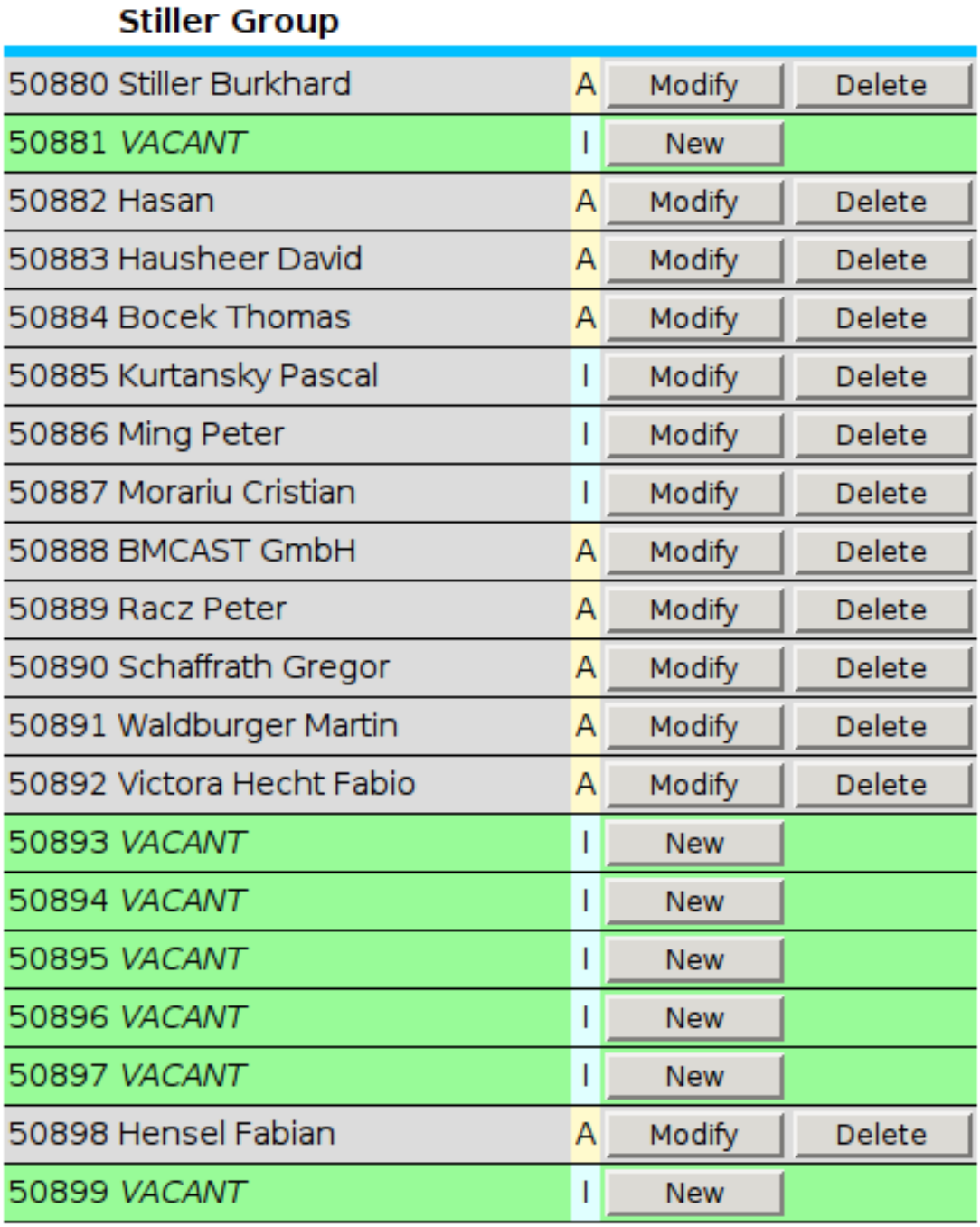
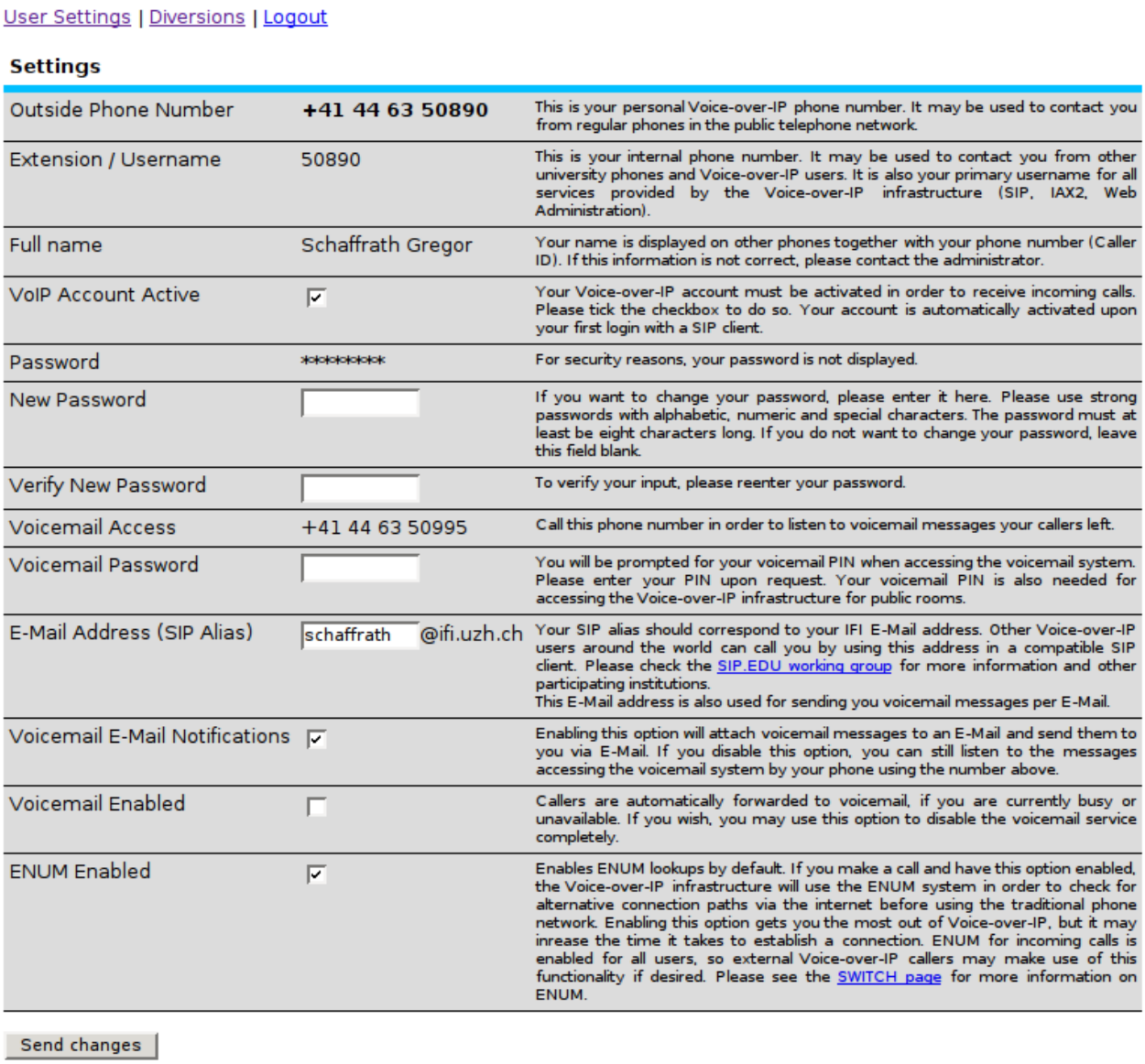


Figure 11: S4E participant administrator view example (the group of Burkhard Stiller in UniZH)

4.4.1 Testing Context

In order to facilitate switches from *stable* to *testing* and back, the context has been designed in a similar fashion to the *stable* context. Furthermore, to avoid collisions between extensions defined by various VoIP research projects, each project receives a three-digit project number to be used in the dialplan. Users and PSTN numbers can be called by prepending a 9 to a phone number in the format of



User Settings | Diversions | Logout

Settings

Outside Phone Number	**+41 44 63 50890**	This is your personal Voice-over-IP phone number. It may be used to contact you from regular phones in the public telephone network.
Extension / Username	50890	This is your internal phone number. It may be used to contact you from other university phones and Voice-over-IP users. It is also your primary username for all services provided by the Voice-over-IP infrastructure (SIP, IAX2, Web Administration).
Full name	Schaffrath Gregor	Your name is displayed on other phones together with your phone number (Caller ID). If this information is not correct, please contact the administrator.
VoIP Account Active	☑	Your Voice-over-IP account must be activated in order to receive incoming calls. Please tick the checkbox to do so. Your account is automatically activated upon your first login with a SIP client.
Password	********	For security reasons, your password is not displayed.
New Password		If you want to change your password, please enter it here. Please use strong passwords with alphabetic, numeric and special characters. The password must at least be eight characters long. If you do not want to change your password, leave this field blank.
Verify New Password		To verify your input, please reenter your password.
Voicemail Access	+41 44 63 50995	Call this phone number in order to listen to voicemail messages your callers left.
Voicemail Password		You will be prompted for your voicemail PIN when accessing the voicemail system. Please enter your PIN upon request. Your voicemail PIN is also needed for accessing the Voice-over-IP infrastructure for public rooms.
E-Mail Address (SIP Alias)	schaffrath @ifi.uzh.ch	Your SIP alias should correspond to your IFI E-Mail address. Other Voice-over-IP users around the world can call you by using this address in a compatible SIP client. Please check the SIP.EDU working group for more information and other participating institutions. This E-Mail address is also used for sending you voicemail messages per E-Mail.
Voicemail E-Mail Notifications	☑	Enabling this option will attach voicemail messages to an E-Mail and send them to you via E-Mail. If you disable this option, you can still listen to the messages accessing the voicemail system by your phone using the number above.
Voicemail Enabled	☐	Callers are automatically forwarded to voicemail, if you are currently busy or unavailable. If you wish, you may use this option to disable the voicemail service completely.
ENUM Enabled	☑	Enables ENUM lookups by default. If you make a call and have this option enabled, the Voice-over-IP infrastructure will use the ENUM system in order to check for alternative connection paths via the internet before using the traditional phone network. Enabling this option gets you the most out of Voice-over-IP, but it may inrease the time it takes to establish a connection. ENUM for incoming calls is enabled for all users, so external Voice-over-IP callers may make use of this functionality if desired. Please see the SWITCH page for more information on ENUM.

Send changes

Figure 12: S4E user configuration details view example

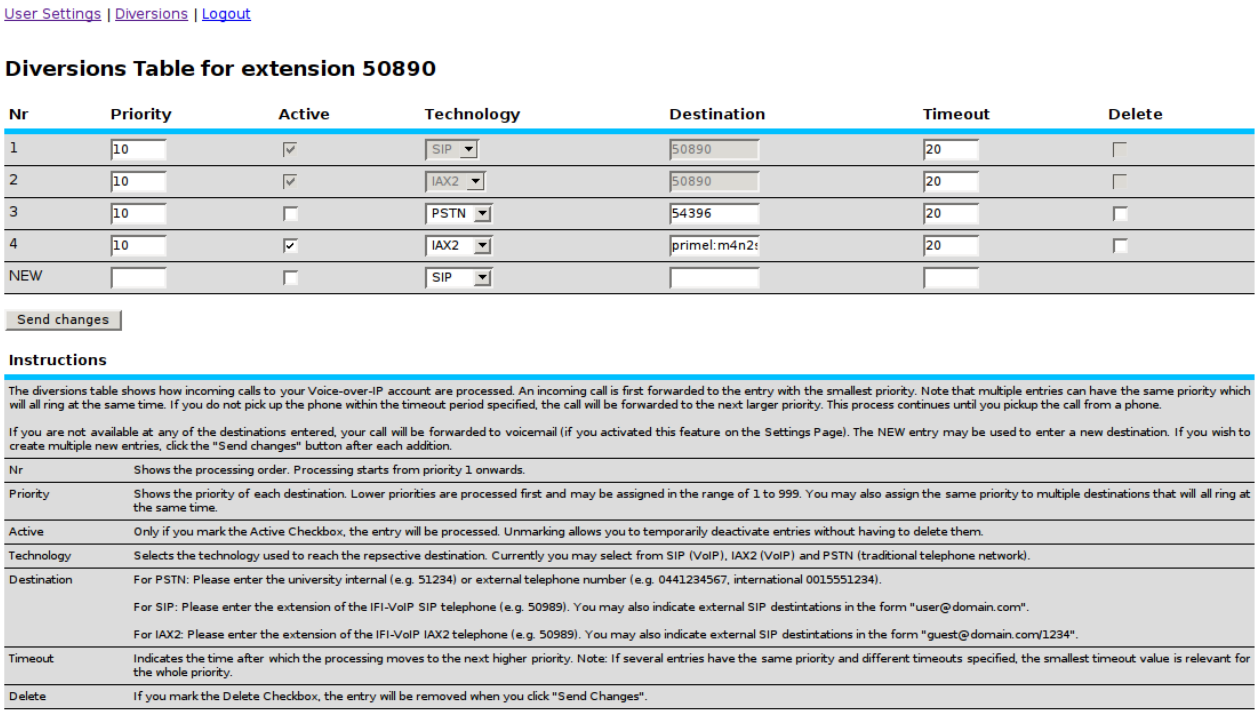
[CountryCode][RegionCode][PhoneNumber]

(*e.g.*, 41446350890). Stable functionality can be accessed by prepending a 8 and thereby explicitly triggering a context switch to the *stable* context. Services can be accessed by dialling 2, followed by the three-digit project number, followed by a four-digit service extension (*e.g.*, 20011000 may be assigned to Service 1000 of VoIP research project 001) [2].

As the testing context is allowed to override *stable* behaviour, this results in a slightly modified behaviour:

- Dialing an extension starting with *0* consults the *testing* diversion database stored in the local MySQL server, before falling back to *stable* procedures.

[2]Stable services are assigned to the project extension '000'



User Settings | Diversions | Logout

Diversions Table for extension 50890

Nr	Priority	Active	Technology	Destination	Timeout	Delete
1	10	☑	SIP	50890	20	☐
2	10	☑	IAX2	50890	20	☐
3	10	☐	PSTN	54396	20	☐
4	10	☑	IAX2	primel:m4n2	20	☐
NEW		☐	SIP			

Send changes

Instructions

The diversions table shows how incoming calls to your Voice-over-IP account are processed. An incoming call is first forwarded to the entry with the smallest priority. Note that multiple entries can have the same priority which will all ring at the same time. If you do not pick up the phone within the timeout period specified, the call will be forwarded to the next larger priority. This process continues until you pickup the call from a phone.

If you are not available at any of the destinations entered, your call will be forwarded to voicemail (if you activated this feature on the Settings Page). The NEW entry may be used to enter a new destination. If you wish to create multiple new entries, click the "Send changes" button after each addition.

Nr	Shows the processing order. Processing starts from priority 1 onwards.
Priority	Shows the priority of each destination. Lower priorities are processed first and may be assigned in the range of 1 to 999. You may also assign the same priority to multiple destinations that will all ring at the same time.
Active	Only if you mark the Active Checkbox, the entry will be processed. Unmarking allows you to temporarily deactivate entries without having to delete them.
Technology	Selects the technology used to reach the repsective destination. Currently you may select from SIP (VoIP), IAX2 (VoIP) and PSTN (traditional telephone network).
Destination	For PSTN: Please enter the university internal (e.g. 51234) or external telephone number (e.g. 0441234567, international 0015551234). For SIP: Please enter the extension of the IFI-VoIP SIP telephone (e.g. 50989). You may also indicate external SIP destintations in the form "user@domain.com". For IAX2: Please enter the extension of the IFI-VoIP IAX2 telephone (e.g. 50989). You may also indicate external SIP destintations in the form "guest@domain.com/1234".
Timeout	Indicates the time after which the processing moves to the next higher priority. Note: If several entries have the same priority and different timeouts specified, the smallest timeout value is relevant for the whole priority.
Delete	If you mark the Delete Checkbox, the entry will be removed when you click "Send Changes".

Figure 13: S4E user diversion details view example

- The same applies to service extensions, although _2000 extensions are by default still reachable in the *testing* context.

4.4.2 Databases

Like the stable environment, the testing environment relies on a table in the MySQL [16] database for diversion information and on a SVN [17] repository for the backup of the context. The SQL database is accessible via the PHPMyAdmin [19] interface.

4.4.3 Experimental Installations

As mentioned before, experimental software and services are not installed on *s4es* and preferably not on any of the connected PSTN gateways participating in the stable environment (*e.g.*, *Partner1* or *Partner2* in Figure 10). Instead, each VoIP research project receives an EMANICSLab slice for experimentation, on which they are able to install arbitrary software and create arbitrary users and contexts. The extensions on these machines are integrated into the S4E environment by placing an appropriate IAX2 or SIP diversion entry into the MySQL table of the *testing* context. User management with respect to access to experimental installations is handled by the EMANICSLab installation.

In case a project requires call permissions on *s4es*, an appropriate user account can be defined in a special *research* group and associated to the *testing* context.

4.5 Call Detail Record Collection

Call Detail Records are collected in the MySQL database on *s4es*. All calls placed by S4E users are registered and made available over the Asterisk-stat interface [20]. This collection includes both calls using S4E features, as well as calls merely looped over *s4es*, as described in the sub-section 4.2.1 to provide realistic usage patterns.

4.6 Distributed Configuration and Storage of User Accounts

A cooperation of different enterprises, institutions or generally speaking organizations always increases the communication needs between organizations. Nowadays a lot of communications is performed via electronic media, like email, wikis or special collaboration tools. Yet, the communication via telephone is still the most important means of communication.

Making a phone call requires knowing the phone number and extension of the callee. Within an organization a corporate directory provides the extension of all co-workers. But this directory is mostly not publicly readable because of privacy and security issues, leaving project partners calling a receptionist and being connected. Some projects might run a project-specific directory, which is most likely not linked with the corporate one and therefore not automatically updated.

This gap might be closed by MetaVoIP. The name MetaVoIP is modelled after the term metadirectory which is a directory service which combines the data from other directories and displays them in a uniform way [21]. MetaVoIP combines data from different private branch exchanges (PBXs) and provides a centralized phone book with some additional user services. The central phone book is automatically built out of the configuration of the local phone systems. The phone systems might be Voice-over-IP based, but this is not a requirement.

The EMANICS partners face the above mentioned problem, too. With the SIP4EMANICS VoIP testbed they have their own phone network at hand - which is only used rarely because of the missing knowledge about phone numbers and extensions.

4.6.1 Design and Architecture of MetaVoIP

MetaVoIP is designed as a centralized application. It gathers data from remote databases, *i.e.*, the user data is stored and maintained locally at each organization. Figure 14 shows the architecture of MetaVoIP. The User GUI and the Application logic are deployed at one server. The User GUI performs all interactions with the user. The Application logic takes commands from the User GUI, processes them and provides results. It further controls the data source connector and the PBX connector.

The data source connector provides an interface to the configuration data stored in each organization. It is used to retrieve information about users, *e.g.*, from a corporate directory. The PBX connector is an interface to the PBX system of each organization. It is used to control the PBX systems, *e.g.*, to initiate a call using the User GUI.

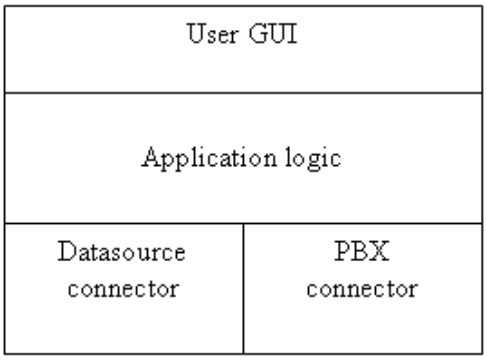


Figure 14: Architecture of MetaVoIP

MetaVoIP topology MetaVoIP abstracts from organizations and describes the topology with sites. For simplicity it is assumed that each organization is located at exactly one site and operates there one data source and one PBX. Mappings must be made, if this assumption does not hold.

An example with three sites is shown in Figure 15. Each site with a data source and a PBX is connected to the MetaVoIP server. The connection to the data sources is established by the data source connector, the connection to the PBXs by the PBX connector.

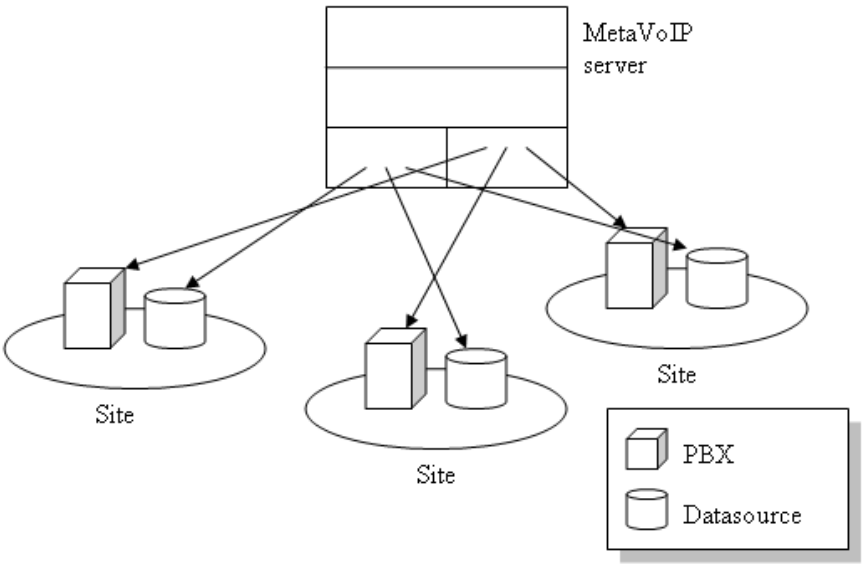


Figure 15: Topology example

MetaVoIP data model MetaVoIP stores a replication of all data read-in from the sites' data sources in an own database. This shortens the access time to this data drastically. In order to keep this data up-to-date, updates are retrieved periodically from the remote data sources.

The data model of MetaVoIP is depicted in Figure 16 in UML format. The class User stores the attributes of a user. Firstname, lastname, email and comments are text fields which

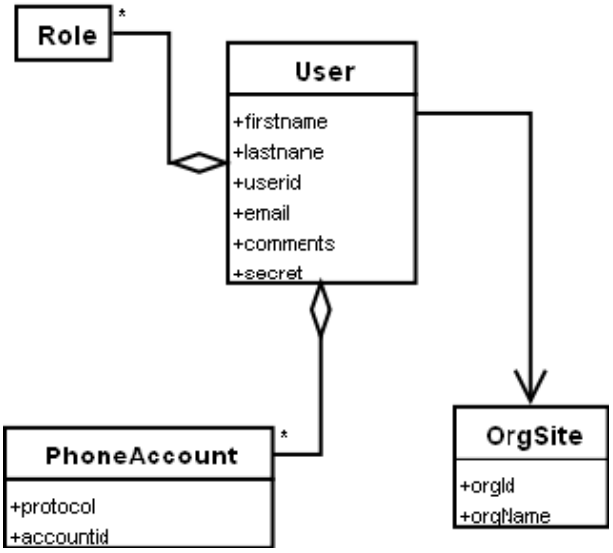


Figure 16: MetaVoIP data model

store the respective information. The field userid stores a site-unique identifier for each user. Secret stores a password which is used for authentication.

The class PhoneAccount models one possibility to call a user. A user might be reachable by different means, *e.g.*, by public switched telephone network (PSTN), Session Inition Protocol (SIP) or Mobile Phone. A description of this means is stored in protocol, in order for the PBX to choose the right output channel when calling the user. Accountid stores the address information, which is protocol dependent, *e.g.*, a phone number for PSTN or a SIP-URL for SIP.

MetaVoIP implements role-based authorization. If a user wants to perform an action in the User GUI, it is checked if he owns the respective role. If not, the user is denied the action. A user might own an arbitrary number of roles.

Each user is member of one site (class OrgSite). A site is identified by a unique identifier (orgid). The attribute orgName is only descriptive. A combination of OrgSite.orgid and User.userid must identify a user unambiguously.

Data source connector The data source connector provides an interface to the user information stored remotely at each site and abstracts from how the data is really stored. A more detailed view of the data source connector is shown in Figure 17. Different data source drivers may be implemented in order to connect to different types of data sources, *e.g.*, LDAP, X.500, SQL database or a driver to retrieve user data from PBXs.

The data source abstraction layer provides a uniform interface between the Application logic and the data source drivers. It is able to choose the right driver depending on the OrgSite.orgid field.

For each site one driver needs to be instantiated. Each driver needs to implement the following functions:

- Read all existing user accounts

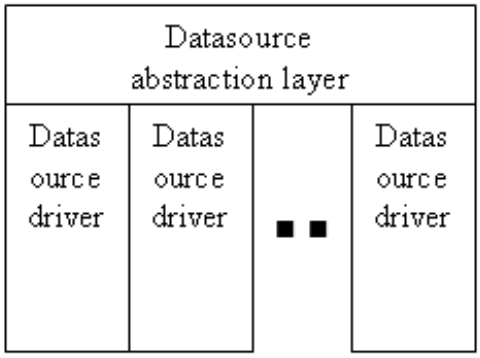


Figure 17: Data source connector

- Read user data for a specified user account
- Read all phone accounts of a user
- Check the password of a user provided by the application logic with the remote data source

If for technical or security reasons it is not possible to read the information remotely a driver might implement a remote agent which is installed at an organization's site, retrieves there the data locally and transfers it to then to the MetaVoIP server.

PBX connector The PBX connector is the interface between the Application logic and the PBX system at a site. Because of the different types of PBX systems and different access possibilities respectively, a PBX connector implements a driver concept, analogously to the data source connector (see Figure 18).

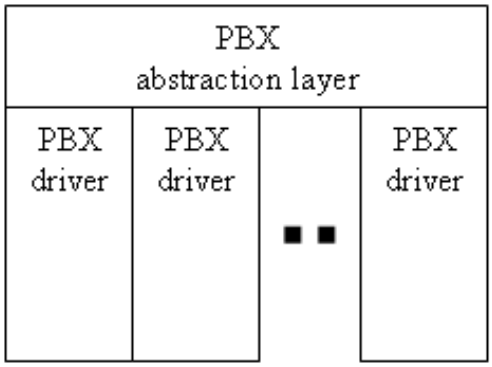


Figure 18: PBX connector

A PBX driver needs to implement the following both functions:

- Check the status of a phone account.
- Create a phone call

The status of a phone should show, if a phone is connected or not. This is of special interest for phones which are not always connected as, *e.g.*, SIP soft phones. In this case showing the state of phone accounts to a user will give him an impression, who is online and reachable and who is not.

A user should be able to select in the User GUI to call somebody else. In this case the remote PBX at the user's site needs to establish a call between the user's phone and the phone of the callee, *i.e.*, create a phone call.

Roles MetaVoIP facilitates a role-based authorization scheme. Table 6 shows the roles and their access profile defined in the application. A user might own several roles at a time.

User	Each user which is listed as a user at any site is automatically assigned the role user.
Usersubadmin	If a user is assigned the role Usersubadmin he is able to manage the accounts of all users, which belong to the same site as him.
Useradmin	Users with the role Useradmin can manage all user account irrespective of the site he belongs to.
Subadmin	This role provides the ability to manage data source drivers and PBX drivers of the own site.
Admin	Admins are able to create, change and delete site definitions, as well as add, change and remove data source drivers and PBX drivers for all sites.

Table 6: Roles in MetaVoIP

MetaVoIP checks the authorization for each action a user wants to perform. If possible, the User GUI displays only those actions, which are allowed for a user. Actions which are not allowed for the logged in user are hidden.

4.6.2 Implementation

MetaVoIP is implemented as a web application using programming language Java and a Apache Jakarta Tomcat Server. This requires also technologies from the Java Enterprise Edition. Here especially Java Server Pages (JSP) in combination with the Java Server Pages Standard Tag Library (JSTL) should be mentioned.

The architecture described above is refined to implement the Model-View-Controller (MVC) concept. Therefore the framework Struts in version 1.3.8 is used.

The data model shown in 4.6.1 is implemented using Java Beans. A bean for User, OrgSite and PhoneAccount is defined. The roles are not represented by a bean, instead the are represented by an enumeration.

Implementation of the data source driver The data source abstraction layer is realized by a Java interface all data source drivers need to implement. Up to now two data source

drivers are implemented. A driver to read the user data from an LDAP server and a driver to read the data from Asterisk configuration files.

The LDAP driver stores roles with objects of type *OrganizationalRole*. The names of the objects have to be the name of the role with the prefix *metavoip*. The object for the role User is therefore *cn=metavoipUser*. All owners of the role have to be listed in the attribute *roleOccupant*.

Astirectory is an extension to the Asterisk PBX, which allows storing SIP and IAX accounts in an LDAP server instead of the respective configuration files. MetaVoIP uses the Astirectory's LDAP schema to store own phone accounts. Astirectory only specifies auxiliary classes, which requires an additional structural class for instantiation. As a structural class in this context *OrganizationalUnit* was chosen. This is a not optimal compromise, but the lesser evil. Extending user objects directly with Astirectory classes is not an option, as one user might have several phone accounts. The phone account classes have to be children of the respective user. Up to now the LDAP driver only supports phone accounts which could be described with the Astirectory classes.

The second driver retrieves its data directly from Asterisk configuration files. The driver periodically downloads the Users, SIP and IAX configuration files from an Asterisk server. The download is performed via Secure Copy (SCP) what requires operating an SSH daemon on the Asterisk server. MetaVoIP then parses these files and extracts the relevant data. The roles a user occupies are stored in the users.conf file as comments of a defined format.

Implementation of the PBX driver The PBX abstraction layer is realized by an abstract Java class, which already implements some common core functionalities. Drivers for a concrete PBX need to override this class and implement all abstract methods.

In general the PBX drivers must implement two main functions:

- establish a call between two named phones
- get the status of a phone account

Up to now only a PBX driver for the Asterisk PBX is implemented. This driver uses the Asterisk Manager API for sending commands to the Asterisk PBX. The implementation uses the Asterisk-Java framework. This framework is a java wrapper library for the text based Asterisk Manager API.

4.6.3 Summary

MetaVoIP is a centralized server application, which can improve the integration between cooperating organizations. Its main focus is to loosely couple the PBX systems of the different organizations in order to making calls between organizations easier, as the effort for looking up phone numbers is drastically reduced.

The application retrieves its data automatically from data sources at the respective sites of an organization. The organization keeps full control over its data. Implemented is furthermore a click-to-call functionality on a web-site.

4.7 Architecture Specification for P2PSIP Overlay

In this section, the architecture and integration of the P2PSIP overlay in the EMANICS VoIP testbed is detailed. With P2PSIP taking lead in the research domain, there are very few works that did analyse the interworking of the P2PSIP network with the traditional SIP network. Thus, this extension work of the VoIP testbed will serve as a platform for the EMANICS partners to carry out research activities related to P2PSIP.

4.7.1 About P2PSIP

P2PSIP can be considered as an extension work of SIP, which addresses the use of distributed resource discovery and management instead of the centralized architecture present in the traditional SIP network. This approach is mainly beneficial in terms of scalability and reliability when compared to single point failure in centralized network. However, this can be achieved at the cost of latency to locate the resource. There are various approaches in the implementation of P2PSIP which are under active research. The details and the ongoing activities (research and standardization) for P2PSIP can be found in [22]. In our P2PSIP overlay integration we use the Bamboo DHT [5], as the underlying P2P network which replaces the resource location functionality of the traditional SIP with the DHT *i.e.*, the P2PSIP overlay network will be created with the DHT at the lower level. However, for the interworking of the P2PSIP network with traditional SIP network we have used a modified SIP proxy server which will be connected to both the P2PSIP overlay network and traditional SP network, acting as a gateway between them.

4.7.2 P2PSIP Architecture Specification in VoIP Testbed

In this section, we detail the P2PSIP architecture deployed in the EMANICS VoIP testbed. The figure 19 shows the architecture of the P2PSIP network with the P2PSIP overlay network and the interworking between the traditional SIP network (Asterisk) in EMANICS VoIP testbed. As discussed in the earlier (distributed and stable) section about the use the EMANICSLab for testing purpose, we have integrated the P2PSIP overlay network in the EMANICSLab. As an initial step, the present integration interconnects to the existing VoIP testbed via the INRIAs Asterisk server. However not restricted to only INRIAs Asterisk server but can be extended to other partners or can be integrated with the centralized user management. The P2PSIP user management is out of scope in the present work and can be an interesting research work that can be carried out from the platform. Also, the present architecture only address the registration and call setup for the P2PSIP clients for the moment and does not address the services like voicemail, conference calls, etc., for the P2PSIP clients, which are available for the traditional SIP networks.

There are two important components in the architecture of P2PSIP in the EMANICS VoIP testbed, one being the P2PSIP overlay network itself and second dealing with the interworking of the P2PSIP network with traditional network (Asterisk). Before we look into, how the P2PSIP and interworking are performed, let us see what are the components or the software that are used for the integration.

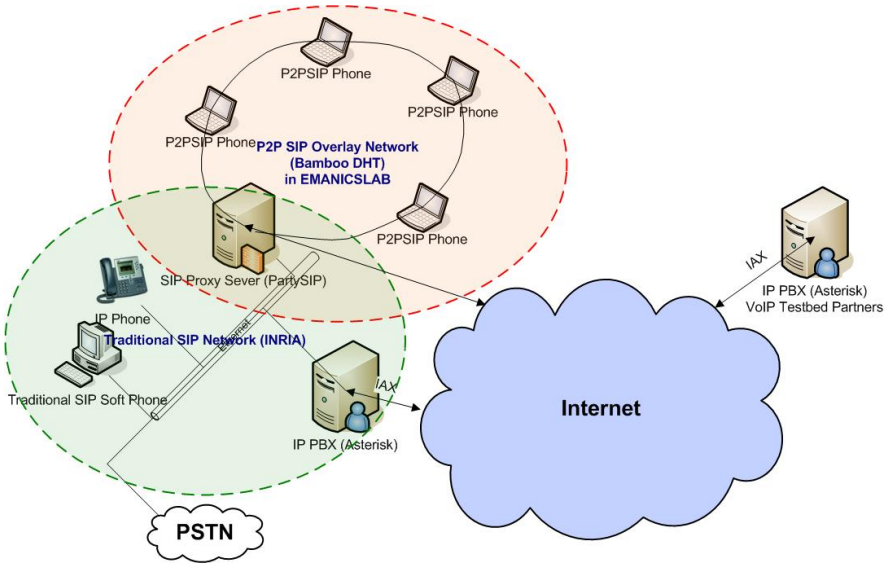


Figure 19: P2PSIP Architecture

1. **P2PSIP Overlay Network:**
   For the integration of the P2PSIP overlay network in the EMANICSLab, we use an open source implementation called Olyo P2PSIP based VoIP System [6]. This P2PSIP implementation uses the Bamboo DHT [5] as the underlying P2P overlay network which serves as a location service for the SIP. The P2PSIP implementation runs on top of P2P overlay making it a P2PSIP overlay network. The details on using this open source software can be found in [6]. A configuration guide is also available.

   The software components that are used from the open source for the setup of the P2PSIP overlay consists of

   - Bamboo DHT(P2P)
   - Modified PartySIP as adapter for SIP clients
   - Any traditional SIP client (Twinkle, kphone, etc.,)

   Since SIP is based on domain based approach, in our architecture P2PSIP overlay network is also considered to be a domain, meaning that each P2PSIP network will have a overlay identifier, which will represent as a domain for the overlay. For example, if we have a P2PSIP overlay with an identity mad1.bamboo and user test1 it can be reached by placing calls as sip:test1@mad1.bamboo.

   The P2PSIP client must first register themselves in the bamboo DHT with user I.D and domain (overlay Identifier). A key will be generated for the client in the DHT. Thus when a call is placed by other clients in the overlay, they will use the same hash function generate the key and query the DHT for routing the calls.

   The dial plan for the P2PSIP overlay clients is straight forward *i.e.*, the P2PSIP overlay is considered to be a single domain or local domain. Therefore when there are calls in the same domain the DHT is used as a location service and calls are placed. Thus, for any client who wish to join the P2PSIP overlay network and make calls,

there no Dial plan specification and just need to register to the DHT. One important issue to note here is that there are no any mechanisms that provide the details of the user I.D that are connected to the P2PSIP overlay network. An illustration of the call flow between two P2PSIP overlay clients is given in Figure 20.

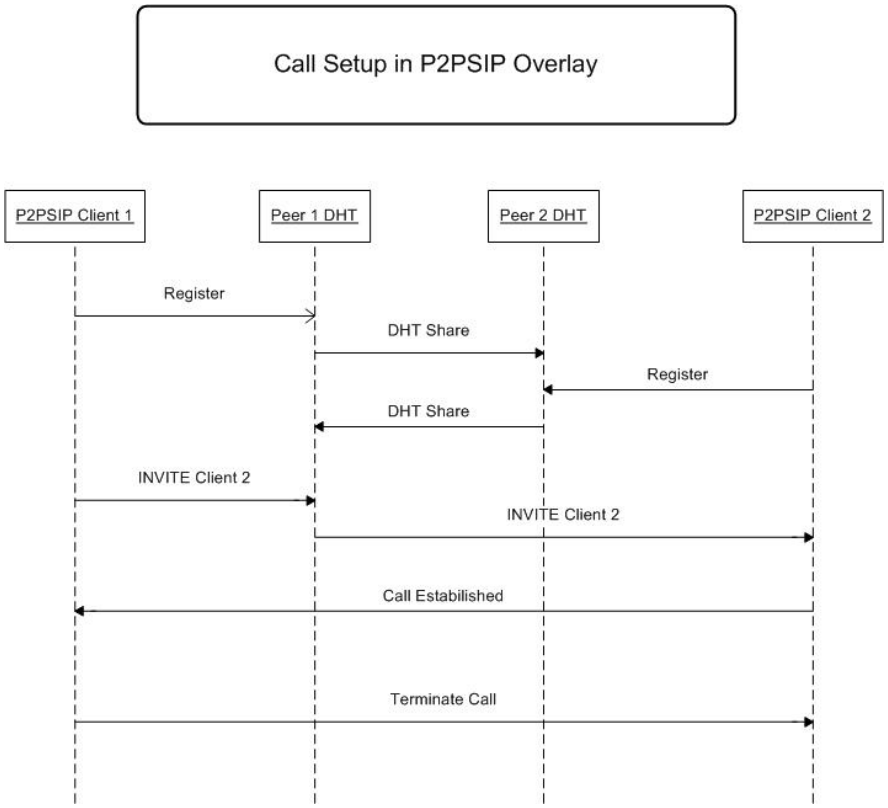


Figure 20: Call Setup between P2PSIP Clients

2. **Interworking of P2PSIP and Traditional SIP (Asterisk):**
   There are very few research works that deal with the interworking of the P2PSIP network with the enterprise level traditional network. In an approach dealing with the interwoking, the P2PSIP overlay is classified as lower and upper level overlay, where the lower level overlay forms the P2PSIP overlay and the higher level overlay serves as a global overlay inter-connecting many P2PSIP overlays. These higher level overlays generally act as the gateway for inter-connecting many heterogeneous overlays. In our case, we use this scenario to inter-connect the traditional SIP network with the P2PSIP network by deploying a SIP proxy server for the traditional network which is also connected to the lower P2PSIP overlay network by creating an upper level overlay.

   The SIP Proxy connects to the P2PSIP overlay by creating a higher level overlay and also joining the lower level overlay. The Proxy server connects to the lower level overlay by adding the upper level overlay ID as the key. Thus when a P2PSIP client makes a call, it looks into the DHT and finds a key for the upper level overlay (SIP Proxy Domain). One important thing to be noted here is that the SIP proxy server only generates a key based on the domain and not with the entire SIP URI part.

The interworking of the P2PSIP and traditional SIP is deployed by a SIP proxy server (Modified PartySIP server olyo implementation) for the Asterisk user, which is eventually connected to the P2PSIP overlay network. The characteristics of the proxy are

- Serves as a traditional SIP proxy for the Asterisk SIP users
- Creates a higher level overlay to serve as gateway for the P2PSIP clients
- Joins the P2PSIP overlay (lower level)

Since we use a SIP proxy server as a gateway for interconnecting the P2PSIP network and traditional SIP network (Asterisk), there are not many changes to be done in the existing dial plan of the Asterisk server in the testbed. Only the architecture of the present testbed is changed by adding a proxy server to route the calls i.e., previously the calls from traditional SIP clients were directly routed by the Asterisk server, but now all route decision for the calls will be made by the proxy server. To accomplish the above scenario some changes in the configuration of the Asterisk in *sip.conf* and *extensions.conf* is needed. Below are the changes done to configuration files in the Asterisk server.

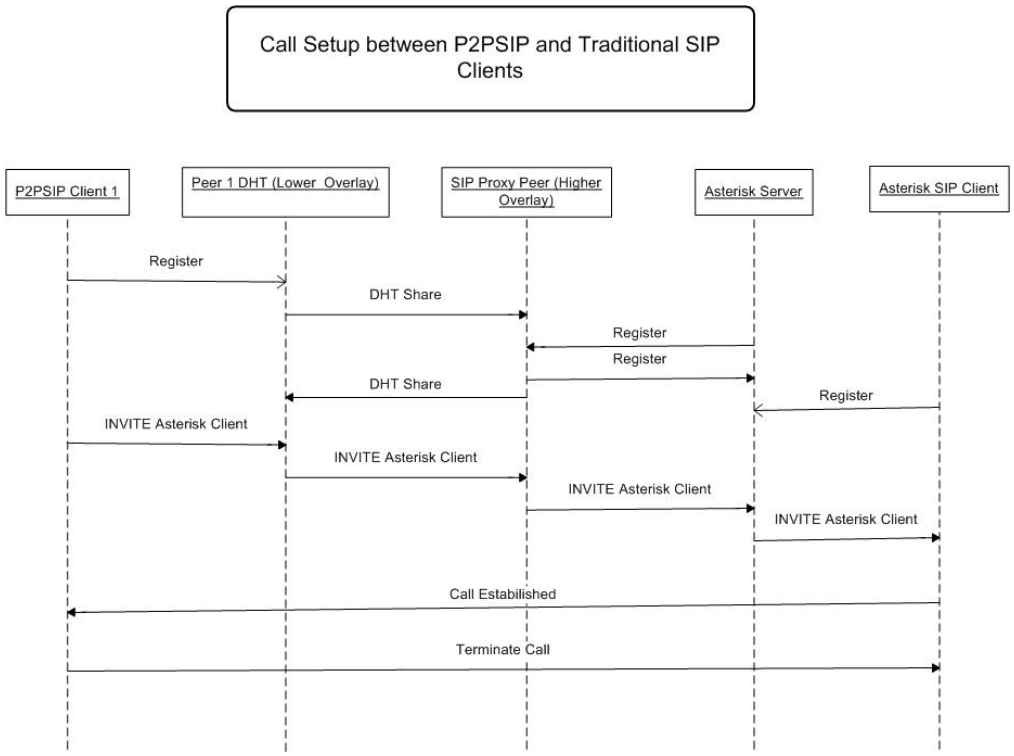


Figure 21: Call Setup from P2PSIP to Traditional SIP

- *sip.conf*
  - Add the details of the SIP proxy server for authorizing the SIP proxy sever
  - Add details to register Asterisk to SIP proxy server

- *extensions.conf*
  - Add context to allow calls from P2PSIP clients via SIP proxy server
  - Add context to allow calls from the SIP clients to P2PSIP clients

To better understand the scenario used in the deployment of P2PSIP with the traditional SIP network can be understood by the call flow diagram between the P2PSIP and traditional SIP clients illustrated in Figures 21 and 22.

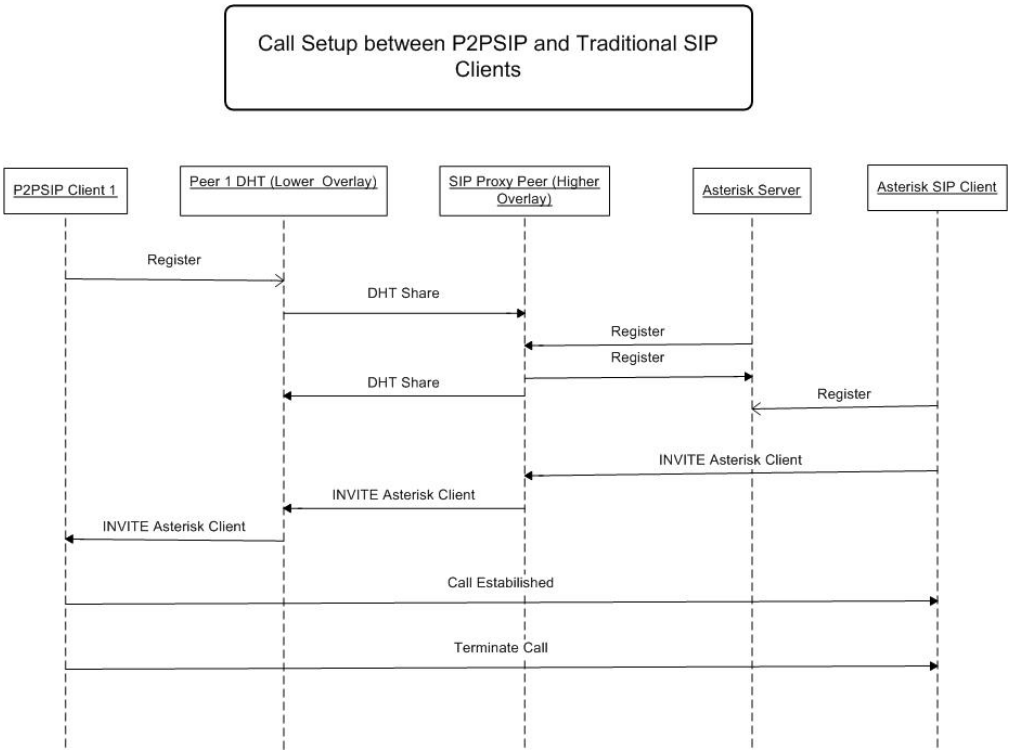


Figure 22: Call Setup from Traditional SIP to P2PSIP Clients

4.8 Nagios Monitoring Architecture for P2PSIP

4.8.1 About Nagios

Nagios is an open source monitoring platform that runs on Linux as a server and periodically launches checks against a group of pre-defined hosts to verify a range of services on each host and to generate a log. Among that, it has an attractive web interface that shows the current global status of the network, and the service or host problems present at each moment. Also, it generates availability or evolution trends in graphs for a given amount of time on a specific host or service. It has built-in support for commonly used protocols.

It executes checks directly or by SNMP, it can monitor resources on the local host and it has the ability to verify services running on remote hosts by SSH connection or using the Nagios Remote Plug-in Executor (NRPE). If the network security policy asks for it, Nagios can even act as a passive agent that receives information from plug-ins that run on the

remote hosts and send results to the Nagios Service Check Acceptor (NSCA), a daemon running along with Nagios. Being aware of the network topology, it has precise localisation of the problems, making the difference between down and unreachable host.

Plug-ins come as separate packages which are downloadable from [23]. A site for additional content (plug-ins, web interface skins, etc) is created by a group of volunteer programmers. These plug-ins are developed according to their specific requirements, like proprietary protocols, industrial appliances or devices not very common on the market.

Usually, plug-ins are Perl script called *check_* and are called by the CGI programs behind the web interface. They return a status value [0 - OK, 1 - WARNING, 2 - CRITICAL, 3 - UNKNOWN)] and performance information, like reply time, bytes received, the server response, etc.

The interaction scheme is given in Figure 23. The plug-in called by the Nagios core runs locally or remotely, and when its execution ends, the response is returned to the web interface.

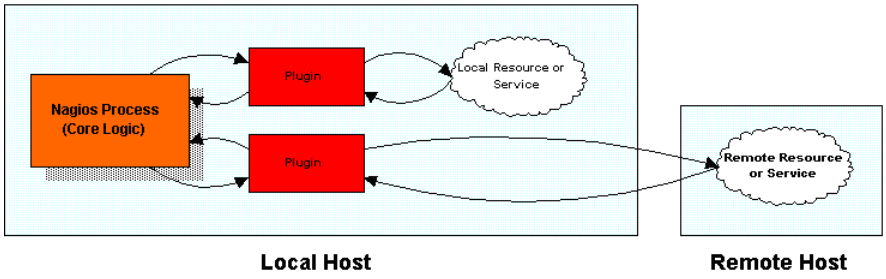


Figure 23: Nagios plug-in working scheme

This information can then be used by an interface to the RRDtool (like NagiosGraph, NagiosGrapher) to draw a graphical representation of the evolution of the status for a host or service on a given amount of time. These graphs are very useful in spotting problems that occur regularly and evaluating trends for a network element activity.

In case of persistent problems, Nagios can automatically execute some tasks (Perl scripts or commands) and send e-mail to operators in charge, escalating with warnings if necessary.

4.8.2 P2P Architecture

P2P systems are decentralized networks of equally-important entities. Each entity is capable of acting both like a client and a server for the others. This type of communication is opposite to the common client/server paradigm, where the server has a known address and it only processes requests made by clients. If the server becomes unreachable, the service becomes unavailable, the clients themselves cannot help each other. Thus, the robustness of the system is seriously affected, having a single point of failure. In a P2P environment, even if some of the peers disappear, the network continues to work, as the remaining peers can communicate with each other. There exist hybrid architectures,

where a server for authentication and registration keeps a global index of peers present online at each moment.

Examples of P2P systems are DC++, eMule, Kazaa, BitTorrent, Gnutella, etc, generally used for file sharing purposes, these networks have evolved strongly in the last years.

Peers contain unique IDs when participating the P2P network. They must know peer IDs to communicate each other. A solution found by the developers is a Distributed Hash Table (DHT), in which every peer has a hash key that identifies it. Each peer also stores information about some of its neighbors to keep in contact if one or more peers go offline. Examples of DHT are Chord, Kademlia (used by eMule), Tapestry, etc. The most important element of DHT is the keyspace, shown in Figure 24.

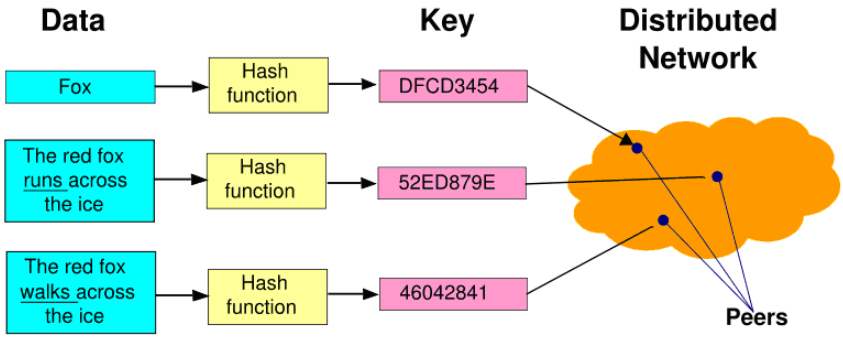


Figure 24: DHT Keyspace

4.8.3 SIP and P2PSIP

The Session Initiation Protocol (SIP) is the signaling text-based protocol used by VoIP devices to negotiate communication before actual voice data is sent over the network. It has a set of messages (INVITE, TRYING, RINGING, etc) that are sent and received between the SIP User Agent (SUA) and the SIP registrar (or proxy), using a typical client/server architecture.

Such architecture has very little scalability because the server has limited resources and can manage only a finite number of requests on a given amount of time. If the number of clients grows dramatically, the computing power needed to handle them becomes expensive and keeping the service alive involves requesting a fee from the clients. That is exactly what some VoIP service providers do, offering low cost voice communications using the IP-based infrastructure instead of expensive telephone lines. But, to offer free and widely distributed service, only a P2P system is suitable.

4.8.4 Bamboo DHT

The creators of Kazaa, using the same P2P principles on which the file-sharing network is based and a proprietary highly-encrypted protocol, have built Skype, the P2P voice

communication system, offered for free but with closed source and protected by industrial patents and strong copyright policy. It is a hybrid system using a set of Global Catalog Servers for authentication and peer discovery, and it uses many NAT-traversal tricks that make it work even with most advanced firewall systems. But, it has no privacy guarantee, as no one knows what the company behind Skype is doing with the traffic if it logs and exploits the conversations. But, this issue is less important for the final home-users, so the system has millions of users connected at every moment of the day.

To build an open-source P2P voice communication system, at the beginning you need a P2P system. The Bamboo DHT package provides a Java-based open-source system based on the Pastry DHT protocols with the advanced features of peer lookup and search. It can also be deployed and integrated in projects with ease. The visual monitoring interface of Bamboo DHT is shown in Figure 25 with peers connected.

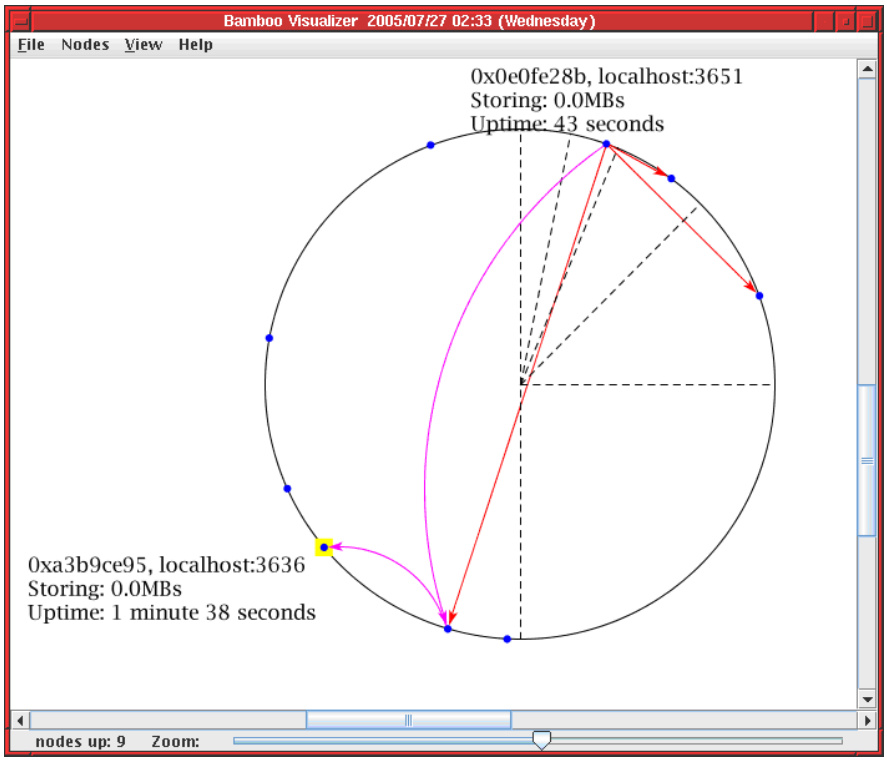


Figure 25: Bamboo DHT Monitoring

4.8.5 Monitoring P2PSIP

Nagios was used for monitoring static infrastructures like a pool of servers. It checks services on hosts by using IP addresses and fixed ports. This information is stored in its configuration files and loaded at the startup of the system. Then, the plug-ins are launched against these hosts and the results are recorded. Particularly, Nagios periodically launches several Perl scripts with parameters (note that operators can do it from the command prompt). The alerting feature, be it by mail, by IM (Jabber) or by other means, is also a command that could be called from a bash script. To monitor remote services,

operators run the plug-in on the server and securely send the results to Nagios, by the NRPE. But, for a P2P client, this isn't an option because not every workstation has SSH and allows a remote machine (like the Nagios server) to connect to it and execute some script, for security reasons.

As the Bamboo system is implemented in Java, the visualization interface offers a good view of peers connected. But, it uses classes and methods from the system core which cannot be called from an independent Perl script. The timing of the checks should not be very tight because the resources required to check a few hundred hosts every minute (for a minimal approach, as P2P networks can be much bigger) would make even the most powerful servers become full loaded. A P2P system is a dynamical infrastructure, where clients dont have static addresses (they usually use ISPs with NAT) and they are hard to be monitored. It is hard to decide if a client has a new address because of disconnection or the client is new. In addition, a client opens a random port for communicating with other peers, not like a server, which has defined ports for the services it offers, but the firewalls used by many clients usually dont accept traffic on every open port.

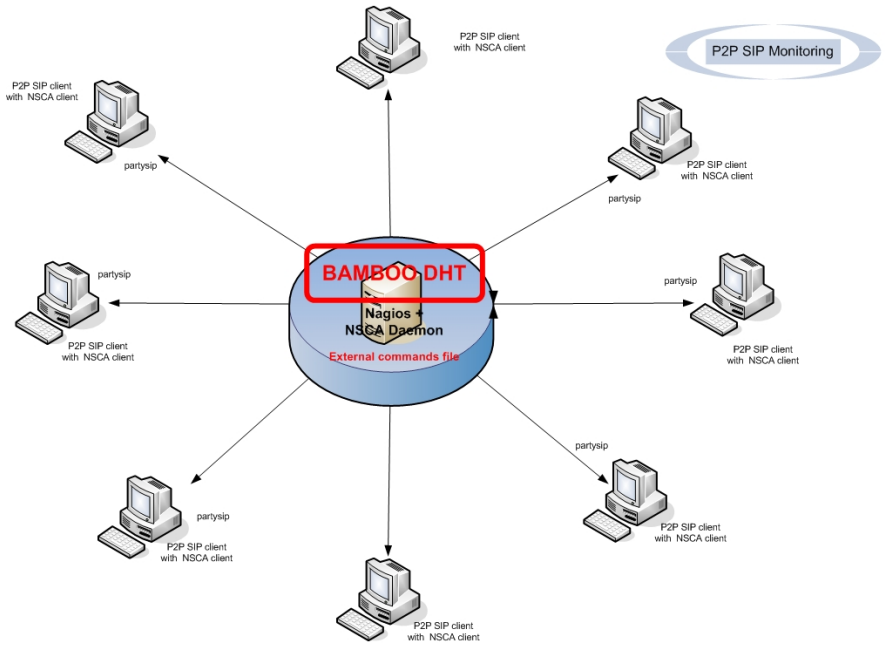


Figure 26: Nagios P2PSIP Monitoring

Considering the mentioned arguments, the only option that can be used to monitor a P2PSIP infrastructure with Nagios is to modify the source code of clients to inform the Nagios server (whos address must be hard-coded or given in a config file) of their presence when they are online. Then, Nagios can verify a client's presence periodically, and when it doesnt receive any answer from the client in a given amount of time, it reports the client's unavailability.

This can be achieved using NSCA, a component of Nagios designed for remote service checks (client binary is send_nsca and config file is ncsa.cfg). The functioning scheme is given in Figure 26.

5 COLLECT

Network traces from operational networks become increasingly important for research and several partners have collected traces on various networks. This project aims at funding network trace data collection activities such as the collection of NETFLOW data sets, the collection of full packet traces, or the collection of network management traffic traces. A collaboration not only yields an impressive set of data traces for current and future research activities but also ensures that the traces are comparable, *i.e.*, collected in similar circumstances, and reusable.

UT has collected NETFLOW data at UT's core network router in order to detect intrusion attempts. One week of flow data has been collected from GEANT (using a 1:1000 sampling), SURFnet (using a 1:100 sampling), and the UT (no sampling). The amount of data collected by UT is several hundreds of GBs. This activity encompassed labeling the data, which has been performed by a honeypot running in parallel inside the UT network. The honeypot information has been collected for one week as well. Some TCP/IP header traces have been made publicly available at http://traces.simpleweb.org.

In addition to network and honeypot information, UT has also collected - together with IUB - SNMP network management traces in order to support research on the analysis of the network management plane. The size of traces collected so far range from 10MB to 130GB. Some SNMP traces have been made available after having sensitive information anonymized. The collected traces comes from 15 different sources, such as university networks, regional network providers, points of presence, and national research backbones.

PSNC is collecting traces from PIONIER NREN. In addition, SNMP network management traces from PIONIER is also collected and provided to EMANICS partners periodically. SNMP traces has been collected over a one month period and the size of the collected information is less than 1 GB. PSNC has also collected and provided partners with sFLOW/NETFLOW traces from PIONIER NREN.

UniZH has collected NETFLOW traces from a local testbed network. These traces have been used for their own research on distributed traffic analysis for flow accounting and intrusion detection. Traces have been collected from PlanetLab, EMANICSLab and the local CSG test bed as well. In addition, UniZH serves as the interface to EMANICSLab by helping to share anonymized traces over the EMANICSLab infrastructure. UniZH has contacted SWITCH in order to get access to NETFLOW traces collected within the Swiss national research network. Currently UniZH together with SWITCH are settling the legal aspects of a collaboration on NETFLOW traces sharing.

LMU has collected NETFLOW traces from the Munich research network at LRZ. These traces are available in EMANICSLab given that necessary access control mechanisms are in place. NETFLOW traces have been captured over a long period of weeks, generating 20 GB per day without employing any sampling strategy. Current traces are already anonymized. Collection and access to traces of SNMP management traffic is currently under investigation.

UPI has collected traces from the UPI network, RoEduNet (the Romanian Educational Network) and or RDS (localnational data&voice ISP). SNMP and Syslog traces have been captured from RoEduNet as well. Traces have been captured for 6, 13, and 8 days,

generating traces whose size range from 330 MB to 2GB. Traces are available after signing an NDA.

Generally, the activity of collecting network traces is running well, including traffic / information from SNMP, SYSLOG, NETFLOW, and TCP/IP header. After March 2008 further activities will be carried out in order to improve the internal availability of traces to EMANICS partners, for example, by defining a common database model for traffic sharing. In addition, part of the collected traces will be possibly made available to researchers outside EMANICS.

6 Collaboration

Collaboration activities among partners are strongly fostered in this work package. Different from the first phase where collaboration activities were more concerned with establishing collaborative infrastructures by integrating partners' resources and joint testing activities by groups of few partners, the collaboration activity of the projects in the second phase tends to exploit infrastructures and resources (e.g., distributed computation and storage capabilities, and collected traces) for joint research projects in other work packages.

Seven partners from the EMANICSLab project contribute resources to establish a distributed computing and storage testbed. The partner Unis participates in the testbed as a user. Due to several advantages of the testbed, eleven (joint) research activities from all partners have considered this testbed as a collaborative environment to do their experiments and to deploy their works. So far this is the first project the collaborates a high number of partners with many usage activities. Future attempts of extending the testbed in term of flexibility and scalability will likely further increase the collaboration of partners.

The COLLECT project has attracted a high number of partners that collaborate for trace collection activities since the first phase. Nevertheless, the project depended more on individual activities or collaborating activities of few partners. With the support of the distributed storage testbed from the EMANICSLab project in the second phase, six partners not only involve collecting trace but also cooperate for making the trace data available to interested researchers. The latter activity allows partners to use the data trace for joint research work, resulting in more contribution and collaboration within the project.

The SIP4EMANICS project presents a close collaboration among four partners in building the enhanced VoIP infrastructure. The project also involves building the P2PSIP overlay on the top of the EMANICSLab testbed, which inspires partners to exploit the overlay for usage and research purposes. A possible joint work is call trace data collection that collaborates partners from the COLLECT project.

The work package has also acquired better collaboration through several face-to-face meetings and exchanges:

- The most recent meeting was at the P2P EMANICS workshop on March 3rd in Zurich, where several joint studies on the EMANICS testbed were presented and the exploitation of the testbed was also discussed.
- A general WP2 meeting was held on January 16th, 2008 in Barcelona to examine the activities of three projects.
- A two-day workshop was organized together with WP7 on November 8-9, 2007 in Enschede to discuss work within the COLLECT project.
- Another meeting was held in October 2007 in Munich to discuss work within the EMANICSLab project.

In addition to these meetings, there were also several smaller meetings between collaborating partners and continued exchanges using email.

7 Conclusions

The fourth "Virtual Laboratory Integration Report" documents the achievements of the three projects sponsored by WP2 during the last nine-month period (July 2007 – March 2008). The achievements of the three funded projects can be summarized as follows:

- A distributed computing and storage testbed (EMANICSLab) has been established successfully. It is already being used by several partners for different research activities. Some of these research activities are themself collaborative projects.
- The Voice over IP testbed created in the first phase of EMANICS has been successfully extended to separate a stable context from a testing context, to support the distributed storage of user account data, and to integrate P2P-based extensions and monitoring extensions. It is worth mentioning that the P2P overlay actually runs on the EMANICSLab infrastructure.
- The network trace collection and trace maintenance activity has been continued in order to increase the number of traces available to EMANICS partners.

WP2 is progressing smoothly and providing an infrastructure that is increasingly used in research activities of the various partners. The level of collaboration in WP2 is very high as can be seen by the number of face-to-face meetings that have taken place in the last nine-month period and joint research activities triggered by the creation of the virtual laboratory and testbed infrastructure.

The next WP2 deliverable is due at the end of the next nine-month in December 2008. An open call for new projects and continuations of existing projects has already been posted and discussions are underway between EMANICS partners to develop strong joint proposals. Given the success of the EMANICSLab project in becoming an important resource in a relatively short time frame, it can be expected that efforts will be undertaken to further develop this joint infrastructure in 2008.

8 Abbreviations

CDR	Call Detail Record
CGI	Common Gateway Interface
DHT	Distributed Hash Table
DUNDi	Distributed Universal Number Discovery
ENUM	Telephone Number Mapping
IAX	Inter-Asterisk eXchange
JSP	Java Server Page
JSTL	Java Standard Tag Library
LDAP	Lightweight Directory Access Protocol
MVC	Model View Controller
NoE	Network of Excellence
NRPE	Nagios Remote Plugin Executor
NSCA	Nagios Service Check Acceptor
PBX	Private Branch Exchange
PSTN	Public Switched Telephone Network
P2P	Peer-to-Peer
RRD	Round-Robin Database
SIP	Session Initiation Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSH	Secure Shell
SUA	SIP User Agent
VoIP	Voice over Internet Protocol

9 Acknowledgement

This deliverable was made possible due to the large and open help of the WP2 Partners of the EMANICS NoE. Many thanks to all of them.

References

[1] MyPLC: A complete PlanetLab Central (PLC) portable installation. Available at `http://www.planet-lab.org/doc/myplc`; Last accessed February 2008.

[2] PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services. Available at `http://www.planet-lab.org/`; Last accessed February 2008.

[3] EmanicsLab Homepage. Available at `https://emanicslab.csg.uzh.ch/`; Last accessed February 2008.

[4] David Hausheer. Ganglia: EmanicsLab Tutorial. EMANICS WP2 Meeting, January 2008. Available at http://emanicslab.csg.uzh.ch/files/emanicslab-tutorial.pdf.

[5] Bamboo DHT. Available at http://www.bamboo-dht.org/; Last accessed February 2008.

[6] Olyo - P2P SIP based VOIP System. Available at http://code.google.com/p/olyo/; Last accessed February 2008.

[7] Open DHT. Available at http://www.opendht.org/; Last accessed February 2008.

[8] Ganglia: EmanicsLab Node Usage Monitor. Available at http://emanicslab.csg.uzh.ch/ganglia/; Last accessed February 2008.

[9] ElabMoni: EmanicsLab Slice Usage Monitor. Available at http://192.41.135.198/~crysm/elabmoni/; Last accessed February 2008.

[10] Asterisk PBX. Available at http://www.asterisk.org/; Last accessed February 2008.

[11] DISA: Direct Inward System Access (Asterisk Application). Available at http://www.voip-info.org/wiki/index.php?page=Asterisk+cmd+DISA; Last accessed February 2008.

[12] MeetMe Conference Application (Asterisk Application). Available at http://www.voip-info.org/wiki-Asterisk+cmd+MeetMe; Last accessed February 2008.

[13] P. Faltstrom and M. Mealling. The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM). RFC 3761. Available at http://www.ietf.org/rfc/rfc3761.txt ; Last accessed February 2008, 2004.

[14] M. Spencer. Distributed Universal Number Discovery (DUNDi). Internet Draft. Available at http://www.dundi.com/dundi.txt; Last accessed February 2008, 2004.

[15] Asterisk RealTime Architecture. Available at http://www.voip-info.org/wiki-Asterisk+RealTime; Last accessed February 2008.

[16] MySQL Database. Available at http://www.mysql.de/; Last accessed February 2008.

[17] Subversion. Available at http://subversion.tigris.org/ ; Last accessed February 2008.

[18] WebMeetMe Conference Management. Available at http://www.webmeetme.com/; Last accessed February 2008.

[19] PHPMyAdmin MySQL Management. Available at http://www.phpmyadmin.net/; Last accessed February 2008.

[20] Asterisk-Stat CDR Analyzer. Available at http://areski.net/asterisk-stat-v2/about.php; Last accessed February 2008.

[21] Alexander Jede. Design and Implementation of a Framework to Integrate Enterprise Asterisk Systems. Master's thesis, University of Federal Armed Forces Munich, 2007.

[22] P2P SIP. Available at http://www.p2p-sip.org/; Last accessed February 2008.

[23] Nagios Plugins and Add Ons Exchange. Available at http://www.nagiosexchange.org/; Last accessed February 2008.